# Coinsult

# Advanced Manual Smart Contract Audit


**Project:** MillTicketCoin
**Website:** http://milliticket.netlify.app/

| Low-risk | Medium-risk | High-risk |
| --- | --- | --- |
| 2 low-risk code issues found | 0 medium-risk code issues found | 0 high-risk code issues found |

**Contract address**
Not deployed on mainnet yet

# 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

**Total Supply:** 1,000,000,000,000
**Total Holders:** 0 (Contract not yet deployed)
**Top 10 holders:**
(Contract not yet deployed)

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

# Source code

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

(Contract not yet deployed)

# Manual Code Review

## 🟢 Low-risk

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

- 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 _transfer(
        address from,
        address to,
        uint256 amount
    ) private {
        require(from != address(0), "ERC20: transfer from the zero
address");
        require(to != address(0), "ERC20: transfer to the zero
address");
        require(amount > 0, "Transfer amount must be greater than
zero");
        if(from != owner() && to != owner())
            require(amount <= _maxTxAmount, "Transfer amount exceeds
the maxTxAmount.");
        uint256 contractTokenBalance = balanceOf(address(this));
        if(contractTokenBalance >= _maxTxAmount)
        {
            contractTokenBalance = _maxTxAmount;
```

```
        }
        bool overMinTokenBalance = contractTokenBalance >=
numTokensSellToAddToLiquidity;
        if (
            overMinTokenBalance &&
            !inSwapAndLiquify &&
            from != uniswapV2Pair &&
            swapAndLiquifyEnabled
        ) {
            contractTokenBalance = numTokensSellToAddToLiquidity;
            swapAndLiquify(contractTokenBalance);
        }
        bool takeFee = true;
        if(_isExcludedFromFee[from] || _isExcludedFromFee[to]){
            takeFee = false;
        }
        _tokenTransfer(from,to,amount,takeFee);
    }
```

- Unused return
    - Note: ignores return value by uniswapV2Router.addLiquidityETH{value: ethAmount}

```
    function addLiquidity(uint256 tokenAmount, uint256 ethAmount)
private {
        _approve(address(this), address(uniswapV2Router), tokenAmount);
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            owner(),
            block.timestamp
        );
    }
```

## 🟡 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

There is some dead-code inside the smart contract. This is code which is never used and should therefore be removed.

**Notes:**

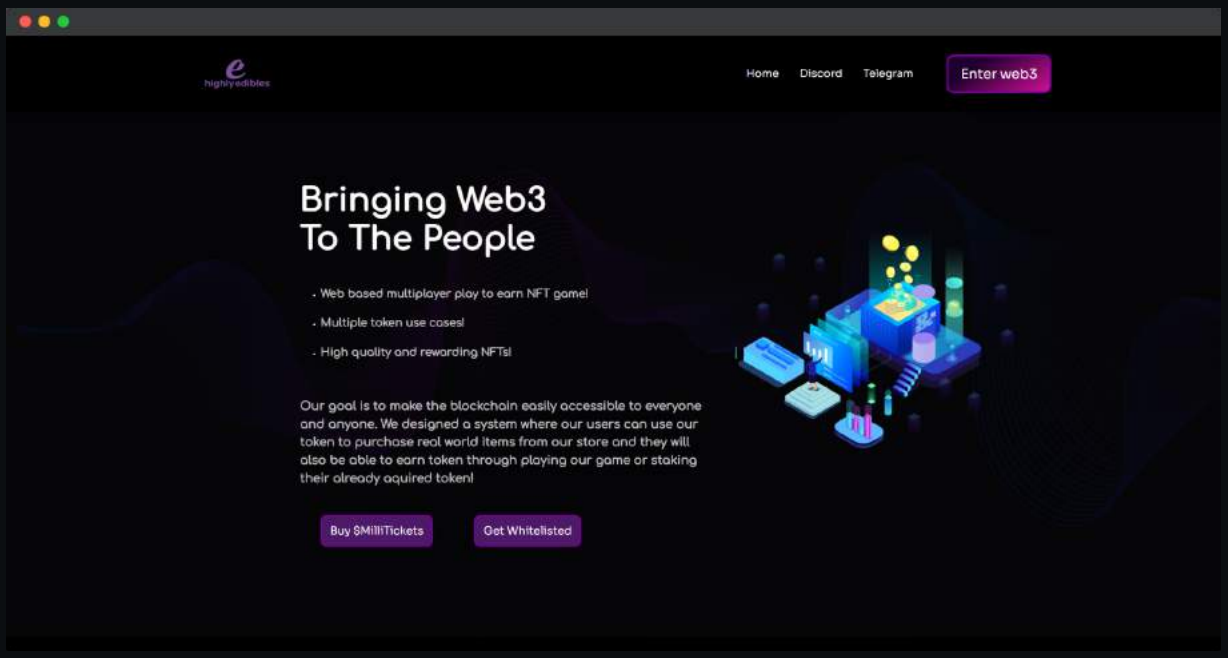
- The owner can set max transaction amount without constraints
- The owner can add or remove any address from fee exclusion at any time
- The ownership of the contract isn’t renounced
- The owner can set fees up to 100%

# Contract Snapshot

```
contract MILLTICKET is Context, IERC20, Ownable {
    using SafeMath for uint256;
    using Address for address;
    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private
_allowances;
    mapping (address => bool) private _isExcludedFromFee;
    mapping (address => bool) private _isExcluded;
    address[] private _excluded;
    address private _developmentWalletAddress =
0xD7C35614A50eCc1d27eC64536A4f5CC5Fc0a9FB9;
    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal = 1000000000000 * 10**18;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));
    uint256 private _tFeeTotal;
    string private _name = "Millticket";
    string private _symbol = "MLT";
    uint8 private _decimals = 18;
    uint256 public _taxFee = 20;
    uint256 private _previousTaxFee = _taxFee;
    uint256 public _developmentFee = 30;
    uint256 private _previousDevelopmentFee = _developmentFee;
    uint256 public _liquidityFee = 50;
    uint256 private _previousLiquidityFee = _liquidityFee;

    IUniswapV2Router02 public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;
    bool inSwapAndLiquify;
    bool public swapAndLiquifyEnabled = true;
    uint256 public _maxTxAmount = 1000000000000 * 10**18;
    uint256 private numTokensSellToAddToLiquidity = 1000000000 *
10**18;
    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
    event SwapAndLiquifyEnabledUpdated(bool enabled);
    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
```

# 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.

🔴 Semi-mobile Friendly (Not optimized for mobile)
🟢 Contains no jQuery errors
🟢 SSL Secured
🟢 Appropriate spelling

**Note: A lot of items are miss-placed on the phone, but the developer told us the website was not yet finished.**

Loading speed: 96%

# Rug-pull Review

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

- Locked Liquidity (no liquidity pool yet)
- Large unlocked wallets (Contract not yet deployed)
- Doxxed team member

# Honeypot Review

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

- Ability to sell
- Owner not able to prevent selling
  Note: fees can be set higher than 25% by owner
- Accurate liquidity pair

**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.