Coinsult

Advanced Manual Smart Contract Audit

Project: Donald John Trump
Website: https://www.djtgame.com/

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
0x41D8675E6C64939c3245908EEaf827A995Fc47Ff

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: 33,333
Total Holders: 2
Top 10 holders:

Rank	Address	Quantity (Token)	Percentage
1	0x6d72fa2b63939cb4367a326f2a86af524578800a	33,333	100.0000%

The top 100 holders collectively own 100.00% (33,333 Tokens) of DJT

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

Source code

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

https://bscscan.com/address/0x41D8675E6C64939c3245908EEaf827A995Fc47Ff#code

Manual Code Review

🟢 Low-risk

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

- Contract contains Reentrancy vulnerabilities:
  uniswapV2Router = _uniswapV2Router
  Additional information: State variables written after the call(s), apply the check-effects-interactions pattern.

```
        // Create a uniswap pair for this new token
        uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());

         uniswapV2Router = _uniswapV2Router;


        emit Transfer(address(0), msg.sender, _tTotal);
```

- Divide before multiply
  Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

```
    function _transferStandard(
        address sender,
        address recipient,
        uint256 tAmount
    ) private {
        _tOwned[sender] = _tOwned[sender].sub(tAmount);

        _takeMarketFee(sender, tAmount.div(10000).mul(_marketFee));

        uint256 recipientRate = 10000 - _marketFee ;
        _tOwned[recipient] = _tOwned[recipient].add(
            tAmount.div(10000).mul(recipientRate)
        );
        emit Transfer(sender, recipient,
tAmount.div(10000).mul(recipientRate));
    }
```

🟡 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 not change fees (1.5% for buy and sell)

🟡 A lot of commented code is in the contract, this could be removed to increase the readability.

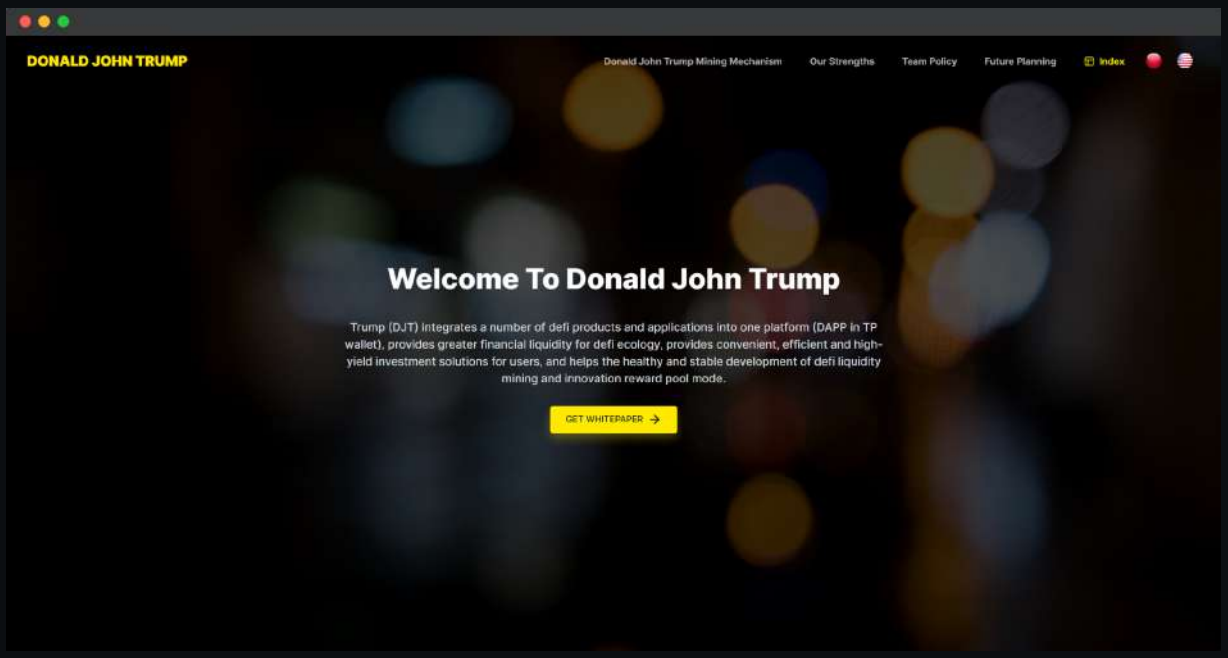
🟡 The ownership is not renounced.

🟡 `SafeMath` is generally not needed starting with Solidity 0.8, since the compiler now has built in overflow checking.

Contract Snapshot

```
contract DJT is IERC20, Ownable {
    using SafeMath for uint256;

    mapping(address => uint256) private _tOwned;
    mapping(address => mapping(address => uint256)) private
_allowances;
    address public marketAddress =
0xdc108b3E92b1F6f36cDaA8069cB20E030f071A3B;
    string private _name = "DJT";
    string private _symbol = "DJT";
    uint8 private _decimals = 9;

    uint256 public _marketFee = 150;
    uint256 private _previoumarketFee;
    uint256 private _tTotal = 33333 * 10**9;
    IUniswapV2Router02 public immutable uniswapV2Router;
    address public immutable uniswapV2Pair;

    constructor() {
        _tOwned[msg.sender] = _tTotal;


        IUniswapV2Router02 _uniswapV2Router = IUniswapV2Router02(
            0x10ED43C718714eb63d5aA57B78B54704E256024E
        );

        // Create a uniswap pair for this new token
        uniswapV2Pair = IUniswapV2Factory(_uniswapV2Router.factory())
            .createPair(address(this), _uniswapV2Router.WETH());

         uniswapV2Router = _uniswapV2Router;


        emit Transfer(address(0), msg.sender, _tTotal);
    }

    function name() public view returns (string memory) {
        return _name;
    }
```

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: 92%

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

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

🟢 Correct router hard coded in the contract

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.