Coinsult

Advanced Manual Smart Contract Audit

Project: Mobipad

Website: https://mobipad.io/

Low-Risk	Medium-Risk	High-Risk
8 low-risk code issues found	3 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x297Cd71a29c987377C7B59Bdf404e926A3A7a3c6

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	MobiPad: Deployer	2	100.0000%

Source Code

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

https://bscscan.com/address/0x297Cd71a29c987377C7B59Bdf404e926A3A7a3c6#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

8 low-risk code
issues found

🟡 Medium-Risk

3 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
) internal {
    if (bpEnabled &amp;amp;&amp;amp; !BPDisabledForever) {
        BP.protect(sender, recipient, amount);
    }

    require(sender != address(0), "BEP20: transfer from the zero address");
    require(recipient != address(0), "BEP20: transfer to the zero address");

    if (_taxEnabled) {
        if (_isIncluded[sender] || _isIncluded[recipient]) {
            // if Sender or Recipient is included in tax collection then implement tax
            _transferWithTax(sender, recipient, amount);
        } else {
            _transferWithoutTax(sender, recipient, amount);
        }
    } else {
        _transferWithoutTax(sender, recipient, amount);
    }
```

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.

Too many digits

Literals with many digits are difficult to read and review.

```
_totalSupply = 2000000000000000000; // 2 MBP initial supply
_maxSupply = 100000000000000000000000000;// 100m max supply
```

Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

Exploit scenario

```
contract MyContract{
    uint 1_ether = 1000000000000000000;
}
```

While `1_ether` looks like `1 ether`, it is `10 ether`. As a result, it's likely to be used incorrectly.

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

No zero address validation for some functions

Detect missing zero address validation.

```
function setTaxStoreAddr(address _addr) external onlyOwner {
    _taxStoreAddr = _addr;
}
```

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 updateBuyTax(uint256 _percentage) external onlyOwner {
    _BUY_TAX = _percentage;
}

function updateSellTax(uint256 _percentage) external onlyOwner {
    _SELL_TAX = _percentage;
}

function setTaxEnabled(bool _enabled) external onlyOwner {
    _taxEnabled = _enabled;
}

function setTaxStoreEnabled(bool _enabled) external onlyOwner {
    _taxStoreEnabled = _enabled;
}

function setTaxStoreAddr(address _addr) external onlyOwner {
    _taxStoreAddr = _addr;
}
```

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.

Boolean equality

Detects the comparison to boolean constants.

```
function setBotProtectionDisableForever() external onlyOwner {
    require(BPDisabledForever == false);
    BPDisabledForever = true;
}
```

Recommendation

Remove the equality to the boolean constant.

Exploit scenario

```
contract A {
    function f(bool x) public {
        // ...
        if (x == true) { // bad!
           // ...
        }
        // ...
    }
}
```

Boolean constants can be used directly and do not need to be compare to `true` or `false`.

🟢 **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.

```
Parameter MobiPad.setBPAddrss(address)._bp (#225) is not in mixedCase
Parameter MobiPad.setBpEnabled(bool)._enabled (#230) is not in mixedCase
Parameter MobiPad.updateBuyTax(uint256)._percentage (#239) is not in mixedCase
Parameter MobiPad.updateSellTax(uint256)._percentage (#243) is not in mixedCase
Parameter MobiPad.setTaxEnabled(bool)._enabled (#247) is not in mixedCase
Parameter MobiPad.setTaxStoreEnabled(bool)._enabled (#251) is not in mixedCase
Parameter MobiPad.setTaxStoreAddr(address)._addr (#255) is not in mixedCase
Parameter MobiPad.setFairsaleContract(address)._fairsaleContract (#542) is not in mixedCase
Parameter MobiPad.mintSupply(address,uint256)._investorAddress (#552) is not in mixedCase
Parameter MobiPad.mintSupply(address,uint256)._amount (#552) is not in mixedCase
Variable MobiPad._BUY_TAX (#188) is not in mixedCase
Variable MobiPad._SELL_TAX (#189) is not in mixedCase
Variable MobiPad.BP (#209) is not in mixedCase
Variable MobiPad.BPDisabledForever (#211) is not in mixedCase
Variable MobiPad.FAIRSALE_CONTRACT (#536) is not in mixedCase
```

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 returns (bytes memory) {
    this;
    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.

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

Costly operations inside a loop

Costly operations inside a loop might waste gas, so optimizations are justified.

```
function excludeAccount(address account) external onlyOwner {
    require(_isIncluded[account], "DPD: Account is already excluded");
    for (uint256 i = 0; i &lt; _included.length; i++) {
        if (_included[i] == account) {
            _included[i] = _included[_included.length - 1];
            _isIncluded[account] = false;
            _included.pop();
            break;
        }
    }
}
```

Recommendation

Use a local variable to hold the loop computation result.

Exploit scenario

```
contract CostlyOperationsInLoop{

    function bad() external{
        for (uint i=0; i < loop_count; i++){
            state_variable++;
        }
    }

    function good() external{
      uint local_variable = state_variable;
      for (uint i=0; i < loop_count; i++){
        local_variable++;
      }
      state_variable = local_variable;
    }
}
```

Incrementing `state_variable` in a loop incurs a lot of gas because of expensive `SSTORE`s, which might lead to an `out-of-gas`.

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Owner can mint new tokens

```
function mint(uint256 amount) public onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    return true;
}
```

Recommendation

Make sure the project owners explain why they need a mint function within their contract.

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Burn can only be called from owner

```
function burn(uint256 amount) external onlyOwner returns (bool) {
    _burn(_msgSender(), amount);
    return true;
}
```

Recommendation

Since the contract uses _msgSender(), you could allow every user to use this function to burn their tokens.

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Spelling error

```
function setBPAddrss(address _bp) external onlyOwner {
    require(address(BP) == address(0), "Can only be initialized once");
    BP = BPContract(_bp);
}
```

Recommendation

change function name to the correct spelling. (setBPAddrss -> setBPAddress)

Owner privileges

- 🟢 Owner cannot pause trading
- 🟢 Owner cannot change max transaction amount
- 🟡 Owner can set fees higher than 25%
- 🟡 Owner can exclude from fees
- 🔴 Owner can mint new tokens
- ⚠️ Owner can set up Bot Prevention

Extra notes by the team

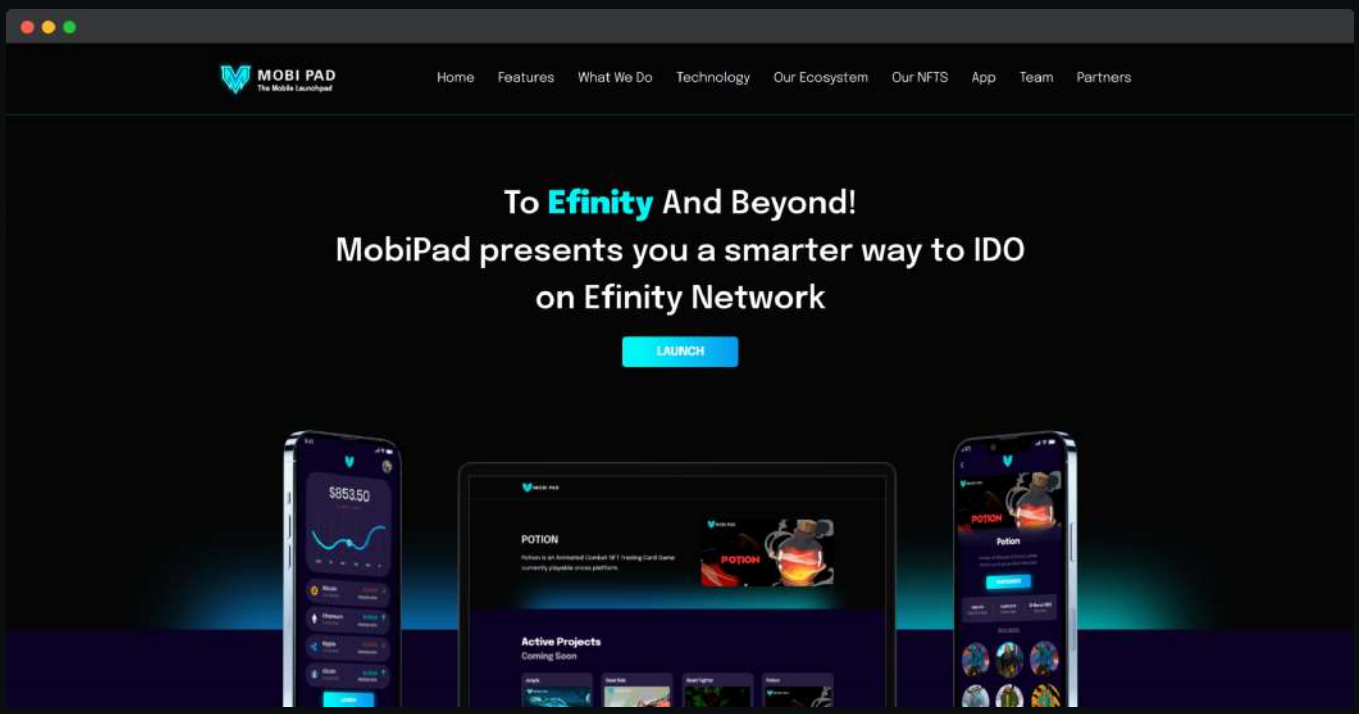
No notes

Contract Snapshot

```
contract MobiPad is Context, IBEP20, Ownable {
using SafeMath for uint256;

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

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

/**
 * @dev Enable or Disable tax system
 */
bool private _taxEnabled;
/**
 * @dev To either store or burn tax
 */
bool private _taxStoreEnabled;

/**
 * @dev tax percentage to be deducted from taxed transaction
 */
uint256 private _BUY_TAX = 1; // %
uint256 private _SELL_TAX = 1; // %

/**
 * @dev address to store tax collected
 */
address private _taxStoreAddr;

/**
 * @dev Store addresses included in tax collection
 * @dev Sending to addresses included in tax collection will remove tax using the percentage in _BUY_
 */
mapping(address =&gt; bool) private _isIncluded;
address[] private _included;

uint256 private _totalSupply;
uint256 private _maxSupply;
uint8 private _decimals;
```

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
- Does not contain jQuery errors
- SSL Secured
- No major spelling errors

Project Overview

KYC verified by Coinsult