Coinsult

Advanced Manual
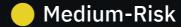# Smart Contract Audit

COIN 1
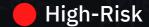
Project: Coin1
Website: Coming Soon

🟢 **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

0xd4023cE8AdDb636BcBdaa07a110A19bb6F93A441

Disclaimer

Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0xe82a03fb7aeb4127b0237a376493e2e7af5f869b	10,000,000	100.0000%

Source Code

Coinsult was comissioned by Coin1 to perform an audit based on the following smart contract:

https://bscscan.com/address/0xd4023ce8addb636bcbdaa07a110a19bb6f93a441#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 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");

    uint256 contractTokenBalance = balanceOf(address(this));

    bool overMinTokenBalance = contractTokenBalance >=
        numTokensSellToAddToLiquidity;
    if (
        overMinTokenBalance &&
        !inSwapAndLiquify &&
        from != uniswapV2Pair &&
        swapAndLiquifyEnabled &&
        _liquidityFee !=0 &&
        _marketingFee !=0
    ) {
        contractTokenBalance = numTokensSellToAddToLiquidity;
```

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.

```
_tTotal = 10000000 * 10**_decimals;
_rTotal = (MAX - (MAX % _tTotal));
numTokensSellToAddToLiquidity = 100000 * 10**_decimals;
```

Recommendation

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

Exploit scenario

```
contract MyContract{
    uint 1_ether = 10000000000000000000;
}
```

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 setMarketingWalletAddress(address _addr) external onlyOwner {
    _marketingWalletAddress = _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.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```solidity
function setSellFeePercent(
    uint256 tFee,
    uint256 lFee,
    uint256 mFee
  ) external onlyOwner {
    _sellTaxFee = tFee;
    _taxFee = _sellTaxFee;
    _sellLiquidityFee = lFee;
    _liquidityFee = _sellLiquidityFee;
    _sellMarketingFee = mFee;
    _marketingFee = _sellMarketingFee;
}

function setBuyFeePercent(
    uint256 tFee,
    uint256 lFee,
    uint256 mFee
) external onlyOwner {
    buyTaxFee = tFee;
```

Recommendation

Emit an event for critical parameter changes.

Exploit scenario

```solidity
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.

```
Variable Ownable._owner (#295) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (#395) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (#397) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (#428) is not in mixedCase
Function IUniswapV2Router01.WETH() (#474) is not in mixedCase
Parameter TOKEN.setMarketingWalletAddress(address)._addr (#985) is not in mixedCase
Parameter TOKEN.setSwapAndLiquifyEnabled(bool)._enabled (#1003) is not in mixedCase
Parameter TOKEN.calculateTaxFee(uint256)._amount (#1128) is not in mixedCase
Parameter TOKEN.calculateLiquidityAndMarketingFee(uint256)._amount (#1132) is not in mixedCase
Variable TOKEN._marketingWalletAddress (#687) is not in mixedCase
Variable TOKEN._buyTaxFee (#697) is not in mixedCase
Variable TOKEN._buyLiquidityFee (#698) is not in mixedCase
Variable TOKEN._buyMarketingFee (#699) is not in mixedCase
Variable TOKEN._sellTaxFee (#701) is not in mixedCase
Variable TOKEN._sellLiquidityFee (#702) is not in mixedCase
Variable TOKEN._sellMarketingFee (#703) 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.

Costly operations inside a loop

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

```
function includeInReward(address account) external onlyOwner {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.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`.

Owner privileges

- 🟢 Owner cannot pause trading

- 🟢 Owner cannot change max transaction amount

- 🟡 Owner can set fees higher than 25%

- 🟡 Owner can exclude from fees

Extra notes by the team

No notes

Contract Snapshot

```solidity
contract TOKEN 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 public _marketingWalletAddress;
address private _burnAddress = 0x000000000000000000000000000000000000dEaD;
uint256 private constant MAX = ~uint256(0);
uint256 private _tTotal;
uint256 private _rTotal;
uint256 private _tFeeTotal;
string private _name;
string private _symbol;
uint256 private _decimals;

uint256 public _buyTaxFee = 5;
uint256 public _buyLiquidityFee = 5;
uint256 public _buyMarketingFee = 5;

uint256 public _sellTaxFee = 5;
uint256 public _sellLiquidityFee = 5;
uint256 public _sellMarketingFee = 5;

uint256 private _taxFee = _buyTaxFee;
uint256 private _liquidityFee = _buyLiquidityFee;
uint256 private _marketingFee = _buyMarketingFee;

uint256 private _previousTaxFee = _taxFee;
uint256 private _previousMarketingFee = _liquidityFee;
uint256 private _previousLiquidityFee = _marketingFee;

IUniswapV2Router02 public uniswapV2Router;
```

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.

🔴 Not Mobile Friendly

🔴 Does contain jQuery errors

🔴 Not SSL Secured

🔴 Contains spelling errors

Note:
Coming soon

Project Overview

● KYC verified by Coinsult

Coin1
Completed KYC Verification at Coinsult.net

Date: 1 July 2022

✔ Project Owner Identified

✔ Contract: 0xd4023cE8AdDb636BcBdaa07a110A19bb6F93A441

Coin1
Audited by Coinsult.net

Date: 1 July 2022

✔ Advanced Manual Smart Contract Audit