Coinsult

Advanced Manual Smart Contract Audit

Project: Divinity

Website: https://www.edendivinity.com

Low-Risk	Medium-Risk	High-Risk
7 low-risk code issues found	1 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x66C26Ac8a7070C06015EbE92A4B90A1D08ae8d13

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	0xee0934691d12dcc4cadd5a2f82ba4d7eeccce2e4	91,188,000,000	56.9925%
2	Null Address: 0x000...dEaD	40,442,513,114	25.2766%
3	0xd47ae1008d51348ae2d4684a372b9c1d2b8d3aa3	9,690,000,000	6.0563%
4	Binance: Hot Wallet 16	1,057,153,416	0.6607%
5	0x2a1ed834b18cd7e022fef91dc2d896a7295832a2	1,052,463,775	0.6578%
6	0x566f42395d1358528d29fda919588cb6f416b662	928,884,000	0.5806%
7	0xc3c9ae9866ebb33bde88aedf0f26b13e37eb384d	928,884,000	0.5806%
8	0xac5f26bb30c9fed45326a42b65251677204b545a	882,943,842	0.5518%
9	0xb57eda267f9b0cb3620f795bf44a9d448fab6766	793,063,326	0.4957%
10	0xb4aafc46cb98acc6303bfb8de33454f4f16cb06d	688,960,525	0.4306%

Source Code

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

https://bscscan.com/address/0x66C26Ac8a7070C06015EbE92A4B90A1D08ae8d13#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

7 low-risk code
issues found

🟡 Medium-Risk

1 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
) internal override {
    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    require(!_isEnemy[from] &amp;&amp; !_isEnemy[to], 'Enemy address');

    if(amount == 0) {
        super._transfer(from, to, 0);
        return;
    }

    uint256 contractTokenBalance = balanceOf(address(this));

    bool canSwap = contractTokenBalance &gt;= swapTokensAtAmount;

    if( canSwap &amp;&amp;
        !swapping &amp;&amp;
        !automatedMarketMakerPairs[from] &amp;&amp;
        from != owner() &amp;&amp;
```

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.

```
secondsUntilAutoClaimAvailable = nextClaimTime &gt; block.timestamp ?
                                                 nextClaimTime.sub(block.timestamp) :
                                                 0;
```

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.

Too many digits

Literals with many digits are difficult to read and review.

```
require(newValue &gt;= 200000 &amp;&amp; newValue &lt;= 500000, &quot;GasForProcessing must be betwe
```

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 setMarketingWallet(address payable wallet) external onlyOwner{
      _marketingWalletAddress = wallet;
  }

function updateUniswapV2Router(address newAddress) public onlyOwner {
      require(newAddress != address(uniswapV2Router), "The router already has that address");
      emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
      uniswapV2Router = IUniswapV2Router02(newAddress);
      address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
          .createPair(address(this), uniswapV2Router.WETH());
      uniswapV2Pair = _uniswapV2Pair;
  }
```

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.

Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
function swapAndSendToFee(uint256 tokens) private  {
    uint256 initialCAKEBalance = IERC20(rewardToken).balanceOf(address(this));
    swapTokensForCake(tokens);
    uint256 newBalance = (IERC20(rewardToken).balanceOf(address(this))).sub(initialCAKEBalance);
    IERC20(rewardToken).transfer(_marketingWalletAddress, newBalance);
    AmountMarketingFee = AmountMarketingFee - tokens;
}
```

Recommendation

Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setBuyTaxes(uint256 liquidity, uint256 rewardsFee, uint256 marketingFee, uint256 deadFee) e
    require(rewardsFee.add(liquidity).add(marketingFee).add(deadFee) &lt;= 25, &quot;Total buy fee i
    buyTokenRewardsFee = rewardsFee;
    buyLiquidityFee = liquidity;
    buyMarketingFee = marketingFee;
    buyDeadFee = deadFee;

}

function setSelTaxes(uint256 liquidity, uint256 rewardsFee, uint256 marketingFee, uint256 deadFee) e
    require(rewardsFee.add(liquidity).add(marketingFee).add(deadFee) &lt;= 25, &quot;Total sel fee i
    sellTokenRewardsFee = rewardsFee;
    sellLiquidityFee = liquidity;
    sellMarketingFee = marketingFee;
    sellDeadFee = deadFee;
}
```

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.

```
Function IUniswapV2Router01.WETH() (#539) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (#703) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (#704) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (#721) 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.

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

Owner can set deadwallet address to any address

```
function setDeadWallet(address addr) public onlyOwner {
        deadWallet = addr;
}
```

Recommendation

Remove this function and hard-code the dead wallet address. If the owner changes this address to a normal address, he can get all the tokens which are meant for the dead wallet.

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
- 🔴 Owner can blacklist addresses
- ⚠️ Owner can set dead wallet address

Extra notes by the team

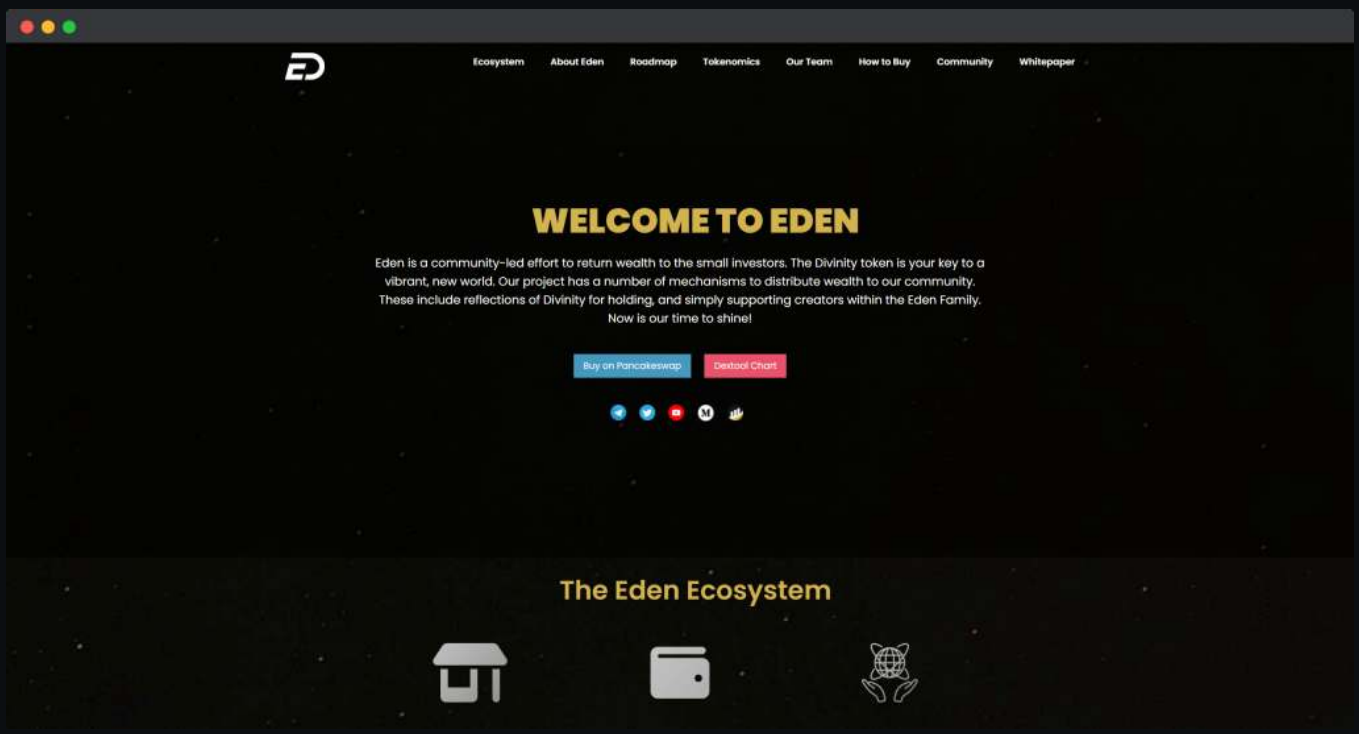
No notes

Contract Snapshot

```
contract CoinToken is ERC20, Ownable {
using SafeMath for uint256;

IUniswapV2Router02 public uniswapV2Router;
address public  uniswapV2Pair;

bool private swapping;

TokenDividendTracker public dividendTracker;

address public rewardToken;

uint256 public swapTokensAtAmount;

uint256 public buyTokenRewardsFee;
uint256 public sellTokenRewardsFee;
uint256 public buyLiquidityFee;
uint256 public sellLiquidityFee;
uint256 public buyMarketingFee;
uint256 public sellMarketingFee;
uint256 public buyDeadFee;
uint256 public sellDeadFee;
uint256 public AmountLiquidityFee;
uint256 public AmountTokenRewardsFee;
uint256 public AmountMarketingFee;

address public _marketingWalletAddress;
address private _node;

address public deadWallet = 0x000000000000000000000000000000000000dEaD;
mapping(address =&gt; bool) public _isEnemy;

uint256 public gasForProcessing;

 // exlcude from fees and max transaction amount
mapping (address =&gt; bool) private _isExcludedFromFees;
```

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