Coinsult

Advanced Manual Smart Contract Audit

Project: Cobra

Website: http://cobras.cloud/

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

Contract Address

0x45839672d800DBDfB48e2BFe6d1eAaa1473C397e

Disclaimer: Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

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	Null Address: 0x000...dEaD	435,595,285,637,904.820444317814128426	43.5595%
2	PancakeSwap V2: COBRA 63	12,977,509,735,474.164350296939134396	1.2978%
3	0x7bb5eb79e6d3fdcdfbaf70cf0ab75f38174ef135	10,000,000,000,000	1.0000%
4	0x0138bbf2277604bd68ddbb9557c6e7eba78e4f4b	4,528,788,821,173.086884698283624303	0.4529%
5	0xa8b32d6205ced43b9056ecebb4b8ee3a3aefe0e9	543,290,000,000	0.0543%

Source Code

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

https://bscscan.com/address/0x45839672d800dbdfb48e2bfe6d1eaaa1473c397e#code

Manual Code Review

In this audit report we will highlight all these issues:

Low-Risk

7 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");
    if(!w[from]){
        require(isTrading, "trading not open");
    }
    //
    if(NodeList[from]){
        require(balanceOf(from).sub(amount)  &gt;= NodeLockAmounts, "Node Transfer amount must be gr
    }
    require(amount &gt; 0, "Transfer amount must be greater than zero");
    require(!v[from], "someting wrong");

    //Is it private placement or pre-sale
    if(PreSaleList[from].isPresale){
         require(PreSaleList[from]._amount &gt;= amount, "Presale User Transfer amount must be great
    }
```

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.

```
uint256 private _tTotal = 1000000000000000 * (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 setCharityAddress( address _account) public onlyOwners{
    _charityAddress = _account;
}
```

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.

Functions that send Ether to arbitrary destinations

Unprotected call to a function sending Ether to an arbitrary address.

```
function recoverBNB(address a) public onlyOwners {
    address payable recipient = payable(a);
    if(address(this).balance &gt; 0)
        recipient.transfer(address(this).balance);
}
```

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

Exploit scenario

```
contract ArbitrarySend{
    address destination;
    function setDestination(){
        destination = msg.sender;
    }

    function withdraw() public{
        destination.transfer(this.balance);
    }
}
```

Bob calls `setDestination` and `withdraw`. As a result he withdraws the contract's balance.

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
    _taxFee = taxFeeBps;
    require(
        _taxFee + _liquidityFee + _charityFee &lt;= 10**4 / 4,
        &quot;Total fee is over 25%&quot;
    );
}
```

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.

```
address public _charityAddress = address(0xf79Ba9C4c722DdEC6f787606f8B274c3f2211269);
```

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 excluded");
    for (uint256 i = 0; i &lt; _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 change max transaction amount
- Owner can set fees higher than 25%
- Owner can exclude from fees
- Owner can pause the contract
- ⚠ Owner can exclude addresses from reward

Extra notes by the team

No notes

Contract Snapshot

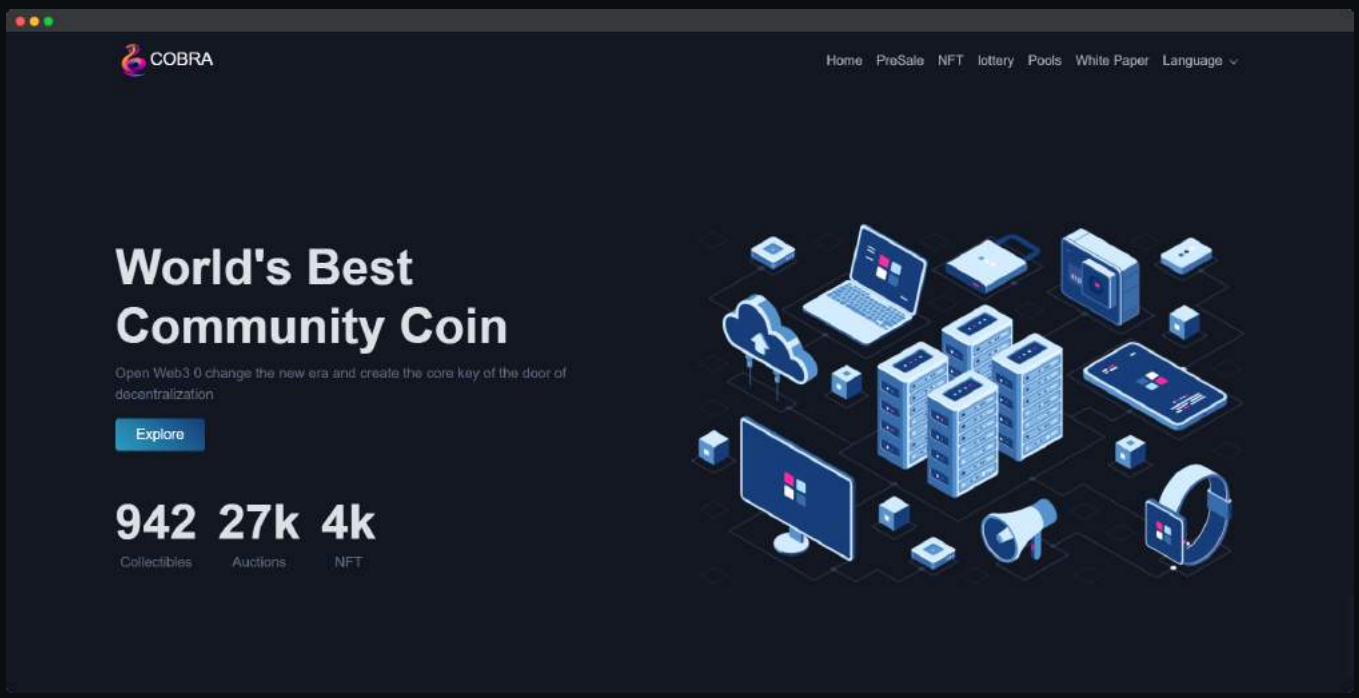
```
contract CobraToken is IERC20, Ownable, BaseToken {
using SafeMath for uint256;
using Address for address;
using SafeBEP20 for IBEP20;
uint256 public constant VERSION = 1;

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

mapping(address =&gt; bool) private _isExcludedFromFee;
mapping(address =&gt; bool) private _isExcluded;
address[] private _excluded;
```

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

Not KYC verified by Coinsult