Coinsult

Advanced Manual Smart Contract Audit

Project: PONTO

Website: https://www.elvenland.com/

🟢 Low-Risk	🟡 Medium-Risk	🔴 High-Risk
6 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract Address

0xe50080AD2699F4fd4DF54b5C8557D06Dc0D2d9d1

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	Elvenland: Deployer	99,993,820.598363999990695999	99.9938%
2	0x9329185d81bc625f8e2bc039c032783f93b83cbe	2,000	0.0020%
3	0x249c965607e9df044e6e7b18d86c23e7a355044c	1,800.0001500000076	0.0018%
4	0x590e0a9940069283770c0bbab7fbc3a588dca044	700	0.0007%
5	0x1312c7d49b2ca7fe89b5dedd258f9053bdcf26c8	514.35	0.0005%

Source Code

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

https://bscscan.com/address/0xe50080AD2699F4fd4DF54b5C8557D06Dc0D2d9d1#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 &gt; 0, "Transfer amount must be greater than zero");
    if(from != owner() &amp;&amp; to != owner())
        require(amount = _maxTxAmount)
    {
        contractTokenBalance = _maxTxAmount;
    }
    bool overMinTokenBalance = contractTokenBalance &gt;= numTokensSellToAddToLiquidity;
    if (
        overMinTokenBalance &amp;&amp;
        !inSwapAndLiquify &amp;&amp;
        from != uniswapV2Pair &amp;&amp;
        swapAndLiquifyEnabled
    ) {
        contractTokenBalance = numTokensSellToAddToLiquidity;
        swapAndLiquify(contractTokenBalance);
```

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 public _maxTxAmount = 1000000000000 * 10**18;
```

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.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
```

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.

```
uint256 public _maxTxAmount = 1000000000000 * 10**18;
```

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 virtual returns (bytes calldata) {
    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 includeInReward(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    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 can change max transaction amount
- Owner can set fees higher than 25%
- Owner can exclude from fees
- Owner can pause the contract

Extra notes by the team

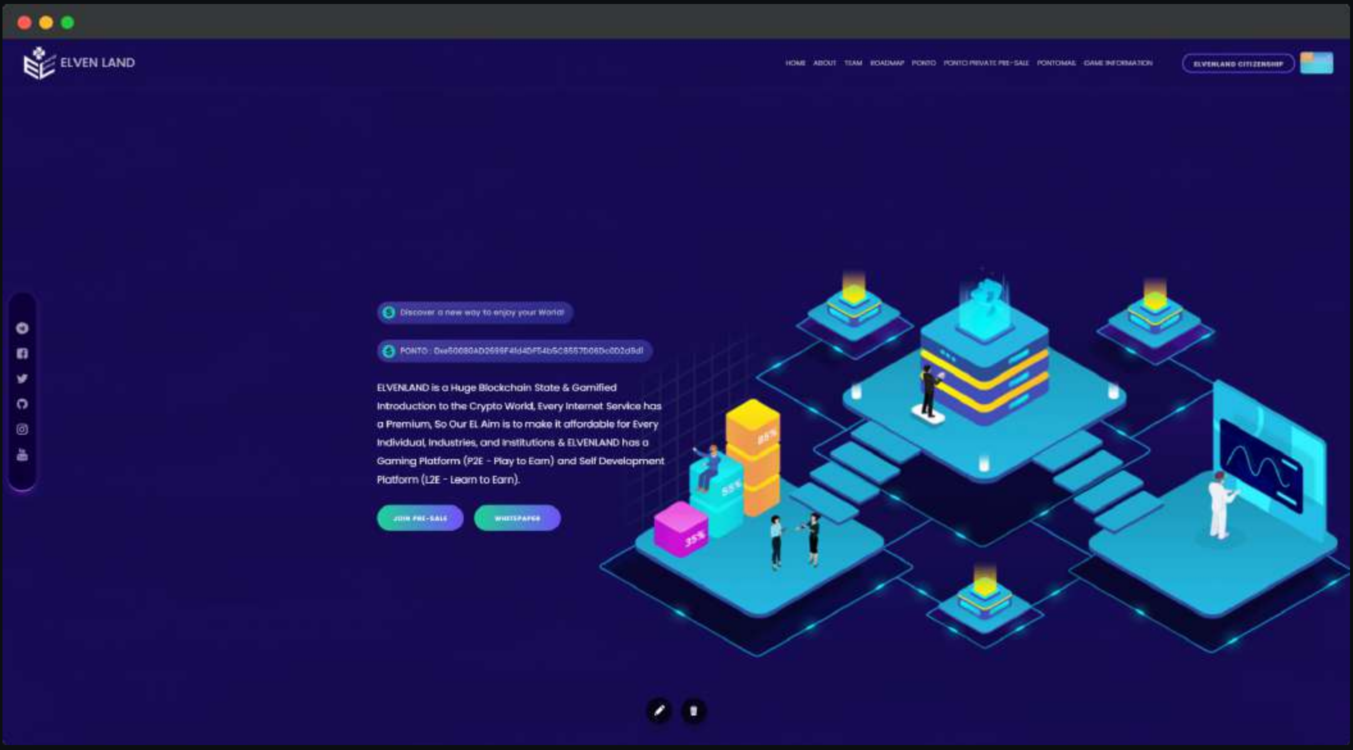
No notes

Contract Snapshot

```
contract PONTOv1 is Context, IERC20, Ownable {
using SafeMath for uint256;
using Address for address;
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;
address private _developmentWalletAddress = 0xEFCa485a6CdCBf2D0E330274133Bb2A790bbeCE5;
uint256 private constant MAX = ~uint256(0);
uint256 private _tTotal = 100000000 * 10**18;
uint256 private _rTotal = (MAX - (MAX % _tTotal));
uint256 private _tFeeTotal;
string private _name = "PONTOv1";
string private _symbol = "PONTO";
uint8 private _decimals = 18;
uint256 public _taxFee = 40;
uint256 private _previousTaxFee = _taxFee;
uint256 public _developmentFee = 30;
uint256 private _previousDevelopmentFee = _developmentFee;
uint256 public _liquidityFee = 50;
uint256 private _previousLiquidityFee = _liquidityFee;
```

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.



Discover a new way to enjoy your World!

PONTO : 0xe50080AD2699F4Id4DF54b5C8557D06Dc0D2d9dI

ELVENLAND is a Huge Blockchain State & Gamified Introduction to the Crypto World, Every Internet Service has a Premium, So Our EL Aim is to make it affordable for Every Individual, Industries, and Institutions & ELVENLAND has a Gaming Platform (P2E - Play to Earn) and Self Development Platform (L2E - Learn to Earn).

- ● Mobile Friendly
- ● Does not contain jQuery errors
- ● SSL Secured
- ● No major spelling errors

Project Overview

● Not KYC verified by Coinsult