Coinsult

Advanced Manual Smart Contract Audit

Project: Ralph

Website: https://ralphtoken.finance/

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

Contract Address

0x7abc725E54e659D0C04875515fE1D21Aa11D44B8

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	696,816,000,000,000	69.6816%
2	0x7252d9c0eae8f0629081dfdc7038620c2ee9e614	233,184,000,000,000	23.3184%
3	0x899fbc12835ef241d5d8beda047546a07438fc07	50,000,000,000,000	5.0000%
4	0x1d14ff9cfbf89941632cc00e7c555106450ac6e9	20,000,000,000,000	2.0000%

Source Code

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

https://bscscan.com/address/0x7abc725e54e659d0c04875515fe1d21aa11d44b8#code

This contract contains imported modules. While Coinsult checks the main contract for vulnarabilities. We can’t guarantee the correctness of these imported modules overtime.

Manual Code Review

In this audit report we will highlight all these issues:

Low-Risk

3 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 sender, address recipient, uint256 amount) private {
    // Make sure that the transaction is not a sandwich pair,
    require(!_isSandwich(sender, recipient, _pairAd));

    // The usual ERC20 checks
    require(sender != address(0), "ERC20: transfer from the zero address");
    require(recipient != address(0), "ERC20: transfer to the zero address");
    require(_balances[sender] &gt;= amount, "ERC20: transfer exceeds balance");
    require(amount &gt; 0, "Transfer = 0");

    // Set defaults for fallback
    uint256 amountRemaining = amount;
    uint256 marketingTax = 0;
    uint256 lotteryTax = 0;
    uint256 burnTax = 0;
    uint256 ralphCapitalTax = 0;
    uint256 autoLiquidityTax = 0;

    // Logic for buys
    if (sender == _pairAd &amp;&amp; recipient != _pancakeRouter &amp;&amp; !_isWhiteListed[recipien
    {
        marketingTax = amount * marketingBuyFee / 100;
```

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.

```
uint256 private _nonce = uint(block.difficulty + block.timestamp);
```

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.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setMarketingBuyFee(uint16 fee) external onlyOwner() {
    marketingBuyFee = fee;
}

function setMarketingSellFee(uint16 fee) external onlyOwner() {
    require(fee + lotterySellFee + burnSellFee + ralphCapitalFee + autoLiquidityFee &lt;= 20, &quot;
    require(fee + lotterySellFee + burnSellFee + whaleRalphCapitalFee + autoLiquidityFee &lt;= 25, &
    marketingSellFee = fee;
}

function setLotteryBuyFee(uint16 fee) external onlyOwner() {
    lotteryBuyFee = fee;
}
```

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.

Owner privileges

- Owner cannot set fees higher than 25%
- Owner cannot pause trading
- Owner cannot change max transaction amount
- Owner can exclude from fees
- ⚠ Owner can set winning lottery numbers
- ⚠ Owner has limited selling fee up to 25% but buying fee has no limit

Extra notes by the team

No notes

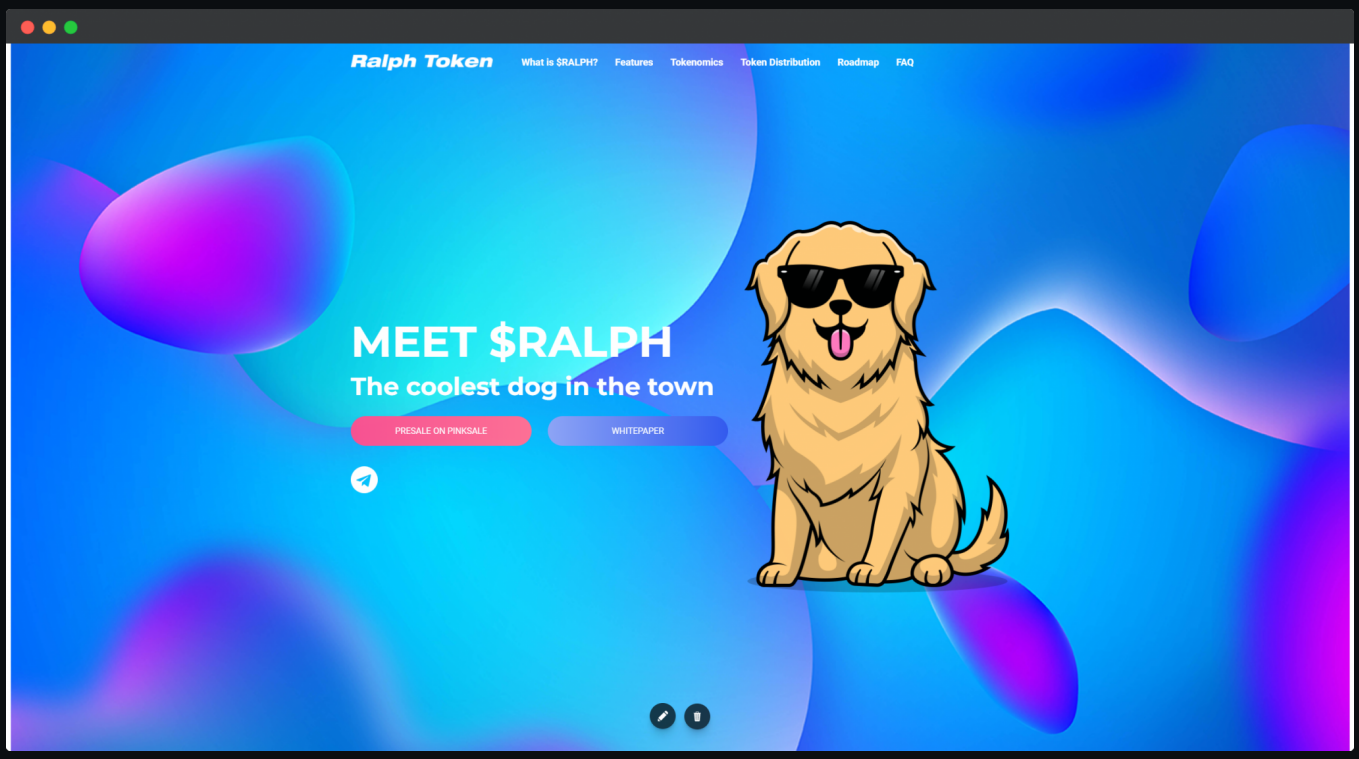
Contract Snapshot

```
contract Ralph is Context, IERC20, Ownable {
// Fee variables
uint16 public marketingBuyFee = 7;
uint16 public marketingSellFee = 5;
uint16 public lotteryBuyFee = 1;
uint16 public lotterySellFee = 1;
uint16 public burnBuyFee = 2;
uint16 public burnSellFee = 2;
uint16 public ralphCapitalFee = 10;
uint16 public whaleRalphCapitalFee = 15;
uint16 public autoLiquidityFee = 2;

// Taxes accumulated in the contract
uint256 private _totalMarketingTax = 0;
```

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