Coinsult

Advanced Manual Smart Contract Audit

Lucky ball

Project: LuckBallClub

Website: https://luckyballclub.github.io/index/

🟢 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

0x3ff3e9a36d49fb2bdbaa767b502e47d6efd12f82

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	0x8db6f0f6374f90b39b74e02989035e1fffb1e90c	318,264,621,413,836	31.8265%
2	0x9b5bd1f818907508484e9692b8493a678361bb5b	199,542,185,771,747	19.9542%
3	PancakeSwap V2: LB-BSC-USD 2	28,957,323,737,868.144536104038794254	2.8957%
4	0xc111c79bac7fa793de3c58f645ce1dd5052f38ed	19,062,740,875,164.498515955753581453	1.9063%
5	0x64801bb22fc3512bbb496a73eb23af1a7cfcdfb0	18,459,606,354,885.832321343934532643	1.8460%

Source Code

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

https://bscscan.com/address/0x3ff3e9a36d49fb2bdbaa767b502e47d6efd12f82#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.

Require return message spelled wrong

```
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

Change “Total sel fee is over 25%” to “Total sell fee is over 25%”

● **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");


    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.

Too many digits

Literals with many digits are difficult to read and review.

```
function updateGasForProcessing(uint256 newValue) public onlyOwner {
    require(newValue &gt;= 200000 &amp;&amp; newValue &lt;= 500000, &quot;GasForProcessing must be b
    require(newValue != gasForProcessing, &quot;Cannot update gasForProcessing to same value&quot;);
    emit GasForProcessingUpdated(newValue, gasForProcessing);
    gasForProcessing = newValue;
}
```

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;
}
```

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 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.

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 exclude addresses from dividends
- ⚠️ Owner can change deadwallet address
- ⚠️ Owner can update claim wait
- ⚠️ Owner can set minimum holding balance to be eligible for dividend

Extra notes by the team

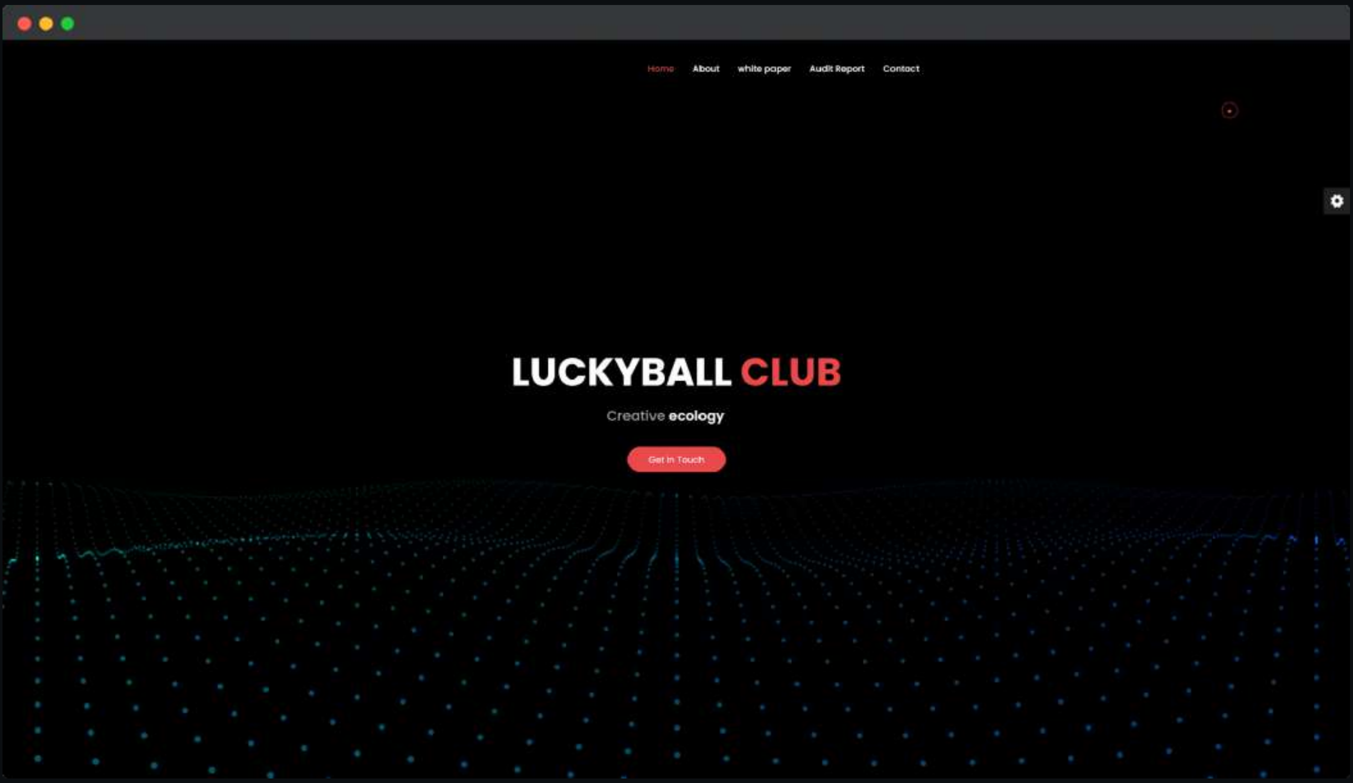
No notes

Contract Snapshot

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

IUniswapV2Router02 public uniswapV2Router;
address public  uniswapPair;

bool private swapping;

BABYTOKENDividendTracker public dividendTracker;

address public rewardToken;

uint256 public swapTokensAtAmount;
```

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

LuckBallClub

Audited by Coinsult.net

Date: 30 August 2022

✓ Advanced Manual Smart Contract Audit