expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

LOL MEME COIN

HIGH RISK ANALYSIS

This section contains a brief Summary of all the High Risks present in this Project's Smart Contract

Findings	Found
High Risk	1

High Risk Details

Owner can set a blacklist account

Risk : High

Overview

Owner can set a blacklist any address and can use it to block any address from trading.

More Details of this High Risk are given on Page 13 of this Audit Report

TABLE OF CONTENTS

OVERVIEW

The Expelee team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analysed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

Audit Result	**Passed with High Risk**
KYC Verification	**No**
Audit Date	**1 May 2023**

CONTRACT DETAILS

Token Name: LOL MEME COIN

Symbol: LOL

Network: Binance Smart Chain

Language: Solidity

Contract Address:
0xF869021097fCd510d509937d54B73c5FBf7e7249

Total Supply: 100000000000

Owner's Wallet:
0x00493146f5A2C5365E9EEB17E0FD614051CC01b2

Deployer's Wallet:
0x00493146f5A2C5365E9EEB17E0FD614051CC01b2

OWNER PRIVILEGES

- Owner can set blacklist any account
- Owner can set fees on buy and sell up to 10%
- Owner can exclude account from fees
- Owner can change swap settings
- Owner can change max tx amount within reasonable limits
- Owner can recover token and bnb from the contract if it is sent there by mistake

AUDIT METHODOLOGY

Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch , that lead to scams and rugpulls.

Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
- Back-doors
- Vulnerability
- Accuracy
- Readability

Tools

- DE
- Open Zeppelin
- Code Analyzer
- Solidity Code
- Compiler
- Hardhat

VULNERABILITY CHECKS

Design Logic	Passed
Compiler warnings	Passed
Private user data leaks	Passed
Timestamps dependence	Passed
Integer overflow and underflow	Passed
Race conditions & reentrancy. Cross-function race conditions	Passed
Possible delays in data delivery	Passed
Oracle calls	Passed
Front Running	Passed
DoS with Revert	Passed
DoS with block gas limit	Passed
Methods execution permissions	Passed
Economy model	Passed
Impact of the exchange rate on the logic	Passed
Malicious event log	Passed
Scoping and declarations	Passed
Uninitialized storage pointers	Passed
Arithmetic accuracy	Passed
Cross-function race conditions	Passed
Safe Zepplin module	Passed

RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and acces control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

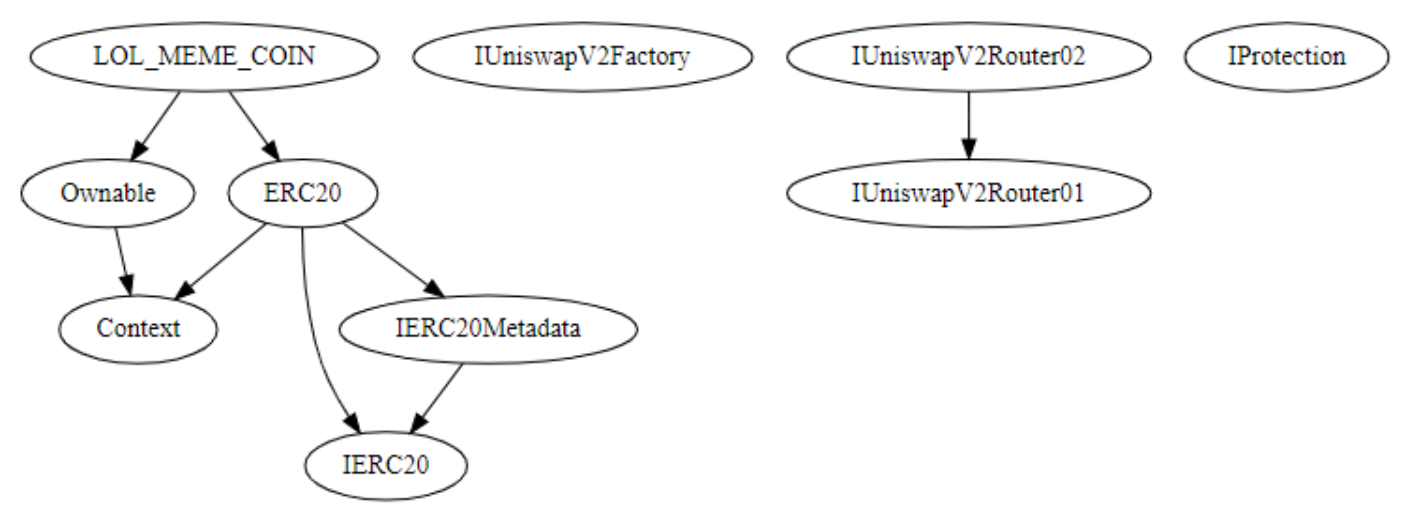
Low Risk

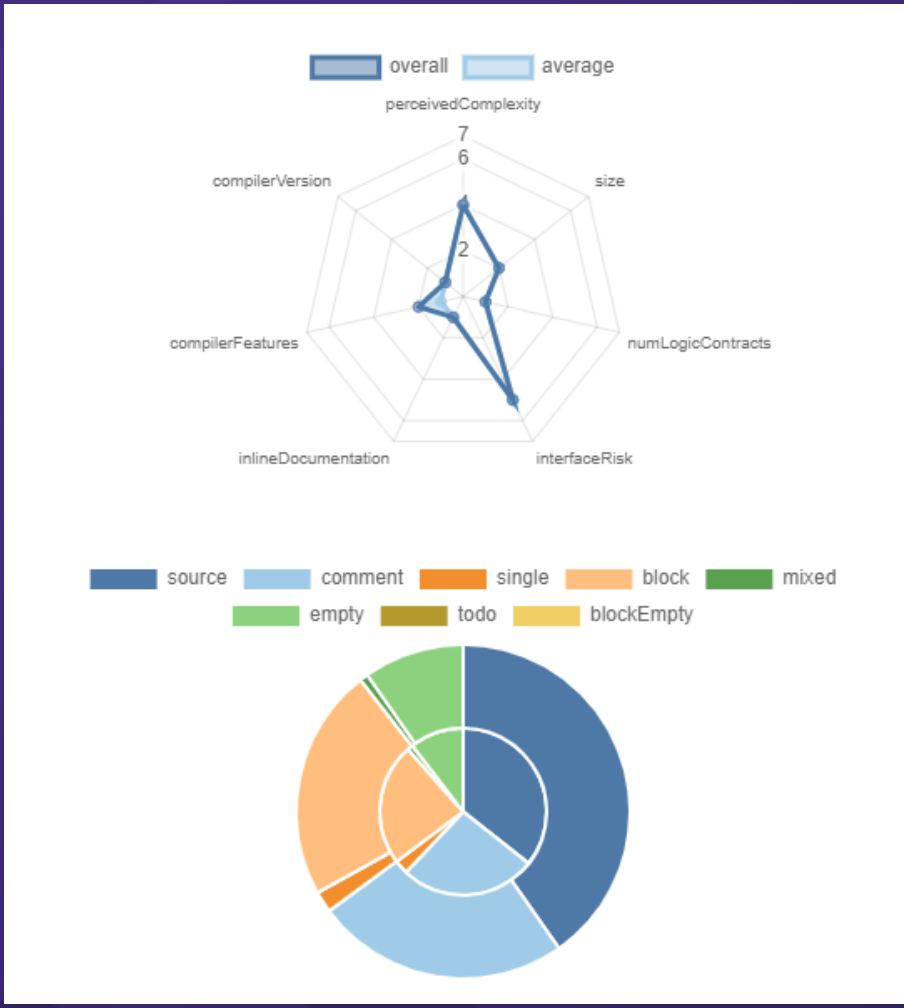
Issues on this level are minor details and warning that can remain unfixed.

Informational

Issues on this level are minor details and warning that can remain unfixed.

INHERITANCE TREES





FUNCTION DETAILS

IProtection	Interface			
└	protection_beforeTokenTransfer	External ❗	🔴	NO ❗
└	isBlacklisted	External ❗		NO ❗
└	isEnabled	External ❗		NO ❗
LOL_MEME_COIN	Implementation	ERC20, Ownable		
└	<Constructor>	Public ❗	🔴	ERC20
└	decimals	Public ❗		NO ❗
└	_transfer	Internal 🔒	🔴	
└	_swapAndSendTreasure	Internal 🔒	🔴	lockTheSwap
└	setExcludedFromFee	External ❗	🔴	onlyOwner
└	setTreasuryFee	External ❗	🔴	onlyOwner
└	setTreasury	External ❗	🔴	onlyOwner
└	setMarketingWallet	External ❗	🔴	onlyOwner
└	setSwapAndTreasureEnabled	External ❗	🔴	onlyOwner
└	setSwapAtAmount	External ❗	🔴	onlyOwner
└	setMaxTxAmountSell	External ❗	🔴	onlyOwner
└	setProtection	External ❗	🔴	onlyOwner
└	recover	External ❗	🔴	onlyOwner
└	revokeBlocked	External ❗	🔴	onlyOwner
└	_beforeTokenTransfer	Internal 🔒	🔴	
└	<Receive Ether>	External ❗	💵	NO ❗
└	emergencyWithdraw	External ❗	🔴	onlyOwner

MANUAL REVIEW

Severity Criteria

Expelee assesses the severity of disclosed vulnerabilities according to methodology based on OWASP standarts.

Vulnerabilities are dividend into three primary risk categroies:
High
Medium
Low

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious input handling
- Escalation of privileges
- Arithmetic
- Gas use

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

FINDINGS

Findings	Severity	Found
High Risk	High	1
Medium Risk	Medium	0
Low Risk	Low	5
Suggestion & discussion	Informational	0
Gas Optimizations	Gas Opt.	0

HIGH RISK FINDING

Owner can set blacklist any account

Severity : High

Overview

Owner can set a blacklist any address and can use it to block any address from trading. Owner has used a bot protection and has blacklist authority thanks to it

```
function _beforeTokenTransfer(
    address from,
    address to,
    uint256 amount
) internal virtual override {
    super._beforeTokenTransfer(from, to, amount);
    if (protection !=  address(0)) {
        IProtection(protection).protection_beforeTokenTransfer(from, to, amount);
    }
}
```

```
  function addToBlacklist(address[] memory users) external onlyOwner {
      for (uint i=0; i < users.length; i++) {
          isBlacklisted[users[i]] = true;
      }
  }
```

Recommendation

Consider implementing an access control mechanism, such as a role-based access control (RBAC) system, to restrict the execution of the **addToBlacklist** function to authorized addresses only. This can help prevent unauthorized individuals from adding addresses to the blacklist.

LOW RISK FINDING

Owner can change buy fees up to 10% sell fees up to 10% at max

Severity : Low

Overview

Functions that allows the owner of the contract to update the buy/sell fees of the contract. These functions assumes that the input parameters are valid and do not exceed the maximum limit of 10% for buy fees and maximum limit of 10% for sell fees.

```
function setTreasuryFee(uint8 _feeOnBuy↑, uint8 _feeOnSell↑) external onlyOwner {
    require(_feeOnBuy↑ <= 10 && _feeOnSell↑ <= 10, 'fee cannot exceed 10%');
    treasuryFeeOnBuy = _feeOnBuy↑;
    treasuryFeeOnSell = _feeOnSell↑;

    emit FeeUpdated(_feeOnBuy↑, _feeOnSell↑);
}
```

Recommendation

It is recommended to add additional access control measures, such as multi-factor authentication or time-based restrictions, to limit the number of authorized users who can call these functions.

LOW RISK FINDING

Owner can exclude accounts from fees

Severity : Low

Overview

Excludes/Includes an address from the collection of fees

```
function setExcludedFromFee(address _account↑, bool _state↑) external onlyOwner {
    require(excludedFromFee[_account↑] != _state↑, 'Already set');
    excludedFromFee[_account↑] = _state↑;
}
```

Recommendation

It is recommended to add additional access control measures, such as multi-factor authentication or time-based restrictions, to limit the number of authorized users who can call these functions. The contract owner account is well secured and only accessible by authorized parties.

LOW RISK FINDING

Owner can change swap setting

Severity : Low

Overview

Function allows the owner of the contract to update the value of **swapAmount**.

```
function setSwapAtAmount(uint256 _amount) external onlyOwner {
    require(_amount > 0, "zero input");
    swapAtAmount = _amount;

    emit SwapAtUpdated(_amount);
}
```

Recommendation

If the threshold is set too low, it could result in frequent and unnecessary swaps, which would increase gas fees and potentially lead to losses due to slippage. On the other hand, if the threshold is set too high, it could result in liquidity being insufficient to handle large trades, which could negatively impact the token price and liquidity pool. Be ensure that the contract owner account is well secured and only accessible by authorized parties.

LOW RISK FINDING

Owner can change max tx amount within reasonable limits

Severity : Low

Overview

Function that allows the contract owner to set the maximum tx amount size of the total token supply.

```
function setMaxTxAmountSell(uint256 _amount↑) external onlyOwner { //@au
    require(_amount↑ >= totalSupply()/1000, "Cannot be less than 0.1%");
    maxTxAmountSell = _amount↑;

    emit MaxSellAmountUpdated(_amount↑);
}
```

Recommendation

While the functions are currently restricted to only be called by the contract owner, it is recommended to consider implementing a more robust access control mechanism.

LOW RISK FINDING

Owner can recover token and bnb from the contract if it is sent there by mistake

Severity : Low

Overview

Owner to withdraw locked or stuck BNB and ERC20 tokens from the contract. The functions are properly restricted to only be executed by the contract owner.

```
function recover(address _token↑, uint256 _amount↑) external onlyOwner { //(
    if (_token↑ != address(0)) {
        IERC20(_token↑).transfer(msg.sender, _amount↑);    Unchecked return
    } else {
        (bool success, ) = payable(msg.sender).call{ value: _amount↑ }("");
        require(success, "Can't send ETH");
    }
}
```

```
function emergencyWithdraw() external onlyOwner {
    (bool success, ) = payable(owner()).call{ value: address(this).balance }('');
    require(success, "Can't send ETH");
}
```

Recommendation

While the functions are currently restricted to only be called by the contract owner, it is recommended to consider implementing a more robust access control mechanism. Also you can add require for can not withdraw native token from the contract.

ABOUT EXPELEE

Expelee is a product-based aspirational Web3 start-up. Coping up with numerous solutions for blockchain security and constructing a Web3 ecosystem from deal making platform to developer hosting open platform, while also developing our own commercial and sustainable blockchain.

www.expelee.com

expeleeofficial
expelee
Expelee
expelee
expelee_official
expelee-co

expelee

Building the Futuristic Blockchain Ecosystem

DISCLAIMER

All the content provided in this document is for general information only and should not be used as financial advice or a reason to buy any investment. Team provides no guarantess against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always do your own research and project yourselves from being scammed. The Expelee team has audited this project for general information and only expresses their opinion based on similar projects and checks from popular diagnostic tools.

Under no circumstances did Expelee receive a payment to manipulate those results or change the awarding badge that we will be adding in our website. Alway do your own research and protect yourselves from scams.

This document should not be presented as a reason to buy or not buy any particular token. The Expelee team disclaims any liability for the resulting losses.