expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Lottery

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	0
Medium	0
Low	2
Informational	2

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**
KYC Verification	**-**
Audit Date	**25 Jan 2024**

CONTRACT DETAILS

Contract Address:
0x4Ea262a96fDE9fD7F298ff59Eac8445aE1Dcc44d

Name: Lottery

Symbol: LUCKY1

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner:
0xB91e327B776BCDa3D7931E4221744F928F796c78

Deployer:
0xB91e327B776BCDa3D7931E4221744F928F796c78

Checksum:
Ede3cef7c2c788bc03532d7342fc9112

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

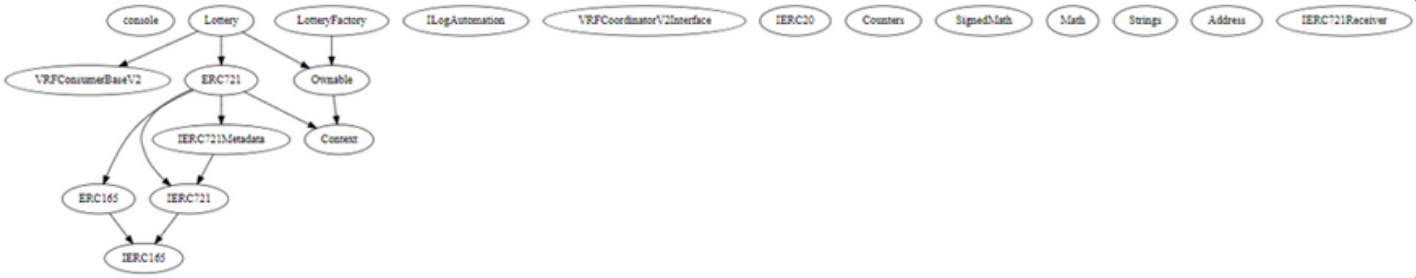
Issues on this level are minor details and warning that can remain unfixed.

Informational

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

expelee

INHERITANCE TREES



POINTS TO NOTE

- The owner can transfer ownership.
- The owner can renounce ownership.
- The owner can Whitelist the Address.
- The owner can set platform fees to not more than 25%.
- The owner can set the creation fee.
- The owner can withdraw the token.

expelee

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in Lottery.purchaseLottery(uint256,address) (Lottery.sol#3898-3959):
        External calls:
        - require(bool,string)(IERC20(feeToken).transferFrom(msg.sender,address(this),totalTicketPrice),Failed to transfer feeToken) (Lottery.sol#3914-3921)
        - require(bool,string)(IERC20(feeToken).transfer(feeAddress,platformCut),Failed to transfer platform fee) (Lottery.sol#3927-3930)
        - IERC20(feeToken).transfer(_affiliateWallet,affiliateCut) (Lottery.sol#3947)
        - _safeMint(msg.sender,tokenId) (Lottery.sol#3953)
                - retval = IERC721Receiver(to).onERC721Received(_msgSender(),from,tokenId,data) (Lottery.sol#3471-3482)
        State variables written after the call(s):
        - myTickets[msg.sender] += 1 (Lottery.sol#3954)
        Lottery.myTickets (Lottery.sol#3845) can be used in cross function reentrancies:
        - Lottery.myTickets (Lottery.sol#3845)
        - Lottery.purchaseLottery(uint256,address) (Lottery.sol#3898-3959)
        - myTicketsIds[msg.sender].push(tokenId) (Lottery.sol#3955)
        Lottery.myTicketsIds (Lottery.sol#3846) can be used in cross function reentrancies:
        - Lottery.myTicketsIds (Lottery.sol#3846)
        - Lottery.purchaseLottery(uint256,address) (Lottery.sol#3898-3959)
        - totalTicketsPurchased ++ (Lottery.sol#3956)
        Lottery.totalTicketsPurchased (Lottery.sol#3835) can be used in cross function reentrancies:
        - Lottery.endLottery(uint32) (Lottery.sol#3965-3972)
        - Lottery.fulfillRandomWords(uint256,uint256[]) (Lottery.sol#4009-4054)
        - Lottery.purchaseLottery(uint256,address) (Lottery.sol#3898-3959)
        - Lottery.requestRandomNumbers(uint32) (Lottery.sol#3982-4003)
        - Lottery.totalTicketsPurchased (Lottery.sol#3835)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
Lottery.fulfillRandomWords(uint256,uint256[]).randomNum (Lottery.sol#4014) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
LotteryFactory.acceptSubscriptionOwnership(uint64) (Lottery.sol#3611-3616) ignores return value by (balance) = vrfCoordinator.getSubscription(_subId) (Lotte
ry.sol#3612)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
Lottery.constructor(string,uint256,uint256,uint256,uint256,uint256[],Lottery.FeeParams,uint64,address)._name (Lottery.sol#3863) shadows:
        - ERC721._name (Lottery.sol#3089) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
LotteryFactory.setAffiliateFeeOnCreate(uint256) (Lottery.sol#3642-3645) should emit an event for:
        - affiliateFeeOnCreation = _affiliateFee (Lottery.sol#3644)
LotteryFactory.setPlatformFee(uint256) (Lottery.sol#3651-3654) should emit an event for:
        - platformFee = _platfomFee (Lottery.sol#3653)
LotteryFactory.setCreationFee(uint256) (Lottery.sol#3656-3658) should emit an event for:
        - creationFee = _creationFee (Lottery.sol#3657)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
LotteryFactory.setPlatformAddress(address)._feeAddress (Lottery.sol#3647) lacks a zero-check on :
                - feeAddress = _feeAddress (Lottery.sol#3648)
LotteryFactory.setUpkeep(address,address)._forwarder (Lottery.sol#3764) lacks a zero-check on :
                - forwarder = _forwarder (Lottery.sol#3766)
Lottery.constructor(string,uint256,uint256,uint256,uint256,uint256[],Lottery.FeeParams,uint64,address)._vrfCoordinator (Lottery.sol#3871) lacks a zero-check
 on :
                - vrfCoordinator = _vrfCoordinator (Lottery.sol#3893)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
ERC721._checkOnERC721Received(address,address,uint256,bytes) (Lottery.sol#3464-3486) has external calls inside a loop: retval = IERC721Receiver(to).onERC721
Received(_msgSender(),from,tokenId,data) (Lottery.sol#3471-3482)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
```

expelee

STATIC ANALYSIS

```
INFO:Detectors:
console._sendLogPayloadImplementation(bytes) (Lottery.sol#12-27) uses assembly
        - INLINE ASM (Lottery.sol#15-26)
console._castToPure(function(bytes)) (Lottery.sol#29-35) uses assembly
        - INLINE ASM (Lottery.sol#32-34)
Math.mulDiv(uint256,uint256,uint256) (Lottery.sol#2081-2160) uses assembly
        - INLINE ASM (Lottery.sol#2088-2092)
        - INLINE ASM (Lottery.sol#2111-2118)
        - INLINE ASM (Lottery.sol#2125-2134)
Strings.toString(uint256) (Lottery.sol#2386-2406) uses assembly
        - INLINE ASM (Lottery.sol#2392-2394)
        - INLINE ASM (Lottery.sol#2398-2400)
Address._revert(bytes,string) (Lottery.sol#2798-2810) uses assembly
        - INLINE ASM (Lottery.sol#2803-2806)
ERC721._checkOnERC721Received(address,address,uint256,bytes) (Lottery.sol#3464-3486) uses assembly
        - INLINE ASM (Lottery.sol#3478-3480)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['>=0.4.22<0.9.0', '^0.8.0', '^0.8.1', '^0.8.4']
        - >=0.4.22<0.9.0 (Lottery.sol#6)
        - ^0.8.0 (Lottery.sol#1703)
        - ^0.8.0 (Lottery.sol#1727)
        - ^0.8.0 (Lottery.sol#1857)
        - ^0.8.0 (Lottery.sol#1938)
        - ^0.8.0 (Lottery.sol#1984)
        - ^0.8.0 (Lottery.sol#2030)
        - ^0.8.0 (Lottery.sol#2372)
        - ^0.8.0 (Lottery.sol#2459)
        - ^0.8.0 (Lottery.sol#2486)
        - ^0.8.0 (Lottery.sol#2818)
        - ^0.8.0 (Lottery.sol#2848)
        - ^0.8.0 (Lottery.sol#2876)
        - ^0.8.0 (Lottery.sol#2907)
        - ^0.8.0 (Lottery.sol#3041)
        - ^0.8.0 (Lottery.sol#3070)
        - ^0.8.0 (Lottery.sol#3536)
```

```
INFO:Detectors:
LotteryFactory.affiliateFee (Lottery.sol#3548) should be constant
LotteryFactory.charity (Lottery.sol#3543) should be constant
LotteryFactory.charityFee (Lottery.sol#3546) should be constant
LotteryFactory.creatorFee (Lottery.sol#3545) should be constant
LotteryFactory.endDate (Lottery.sol#3542) should be constant
LotteryFactory.feeToken (Lottery.sol#3544) should be constant
LotteryFactory.maxTickets (Lottery.sol#3541) should be constant
LotteryFactory.name (Lottery.sol#3538) should be constant
LotteryFactory.startTime (Lottery.sol#3549) should be constant
LotteryFactory.symbol (Lottery.sol#3539) should be constant
LotteryFactory.ticketPrice (Lottery.sol#3540) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
Lottery.affiliateFee (Lottery.sol#3829) should be immutable
Lottery.charity (Lottery.sol#3824) should be immutable
Lottery.charityFee (Lottery.sol#3828) should be immutable
Lottery.creatorFee (Lottery.sol#3827) should be immutable
Lottery.endDate (Lottery.sol#3823) should be immutable
Lottery.factory (Lottery.sol#3838) should be immutable
Lottery.feeAddress (Lottery.sol#3825) should be immutable
Lottery.feeToken (Lottery.sol#3826) should be immutable
Lottery.maxTickets (Lottery.sol#3822) should be immutable
Lottery.maxTicketsPerWallet (Lottery.sol#3831) should be immutable
Lottery.maxWinners (Lottery.sol#3833) should be immutable
Lottery.platformFee (Lottery.sol#3834) should be immutable
Lottery.startTime (Lottery.sol#3830) should be immutable
Lottery.subId (Lottery.sol#3839) should be immutable
Lottery.ticketPrice (Lottery.sol#3821) should be immutable
Lottery.vrfCoordinator (Lottery.sol#3836) should be immutable
LotteryFactory.vrfCoordinator (Lottery.sol#3566) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:Lottery.sol analyzed (20 contracts with 93 detectors), 554 result(s) found
```

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			

LOW RISK FINDING

Category: Centralization
Subject: Missing Events
Status: Open
Severity: Low
Impact:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
function setAffiliateFeeOnCreate(uint _affiliateFee) external onlyOwner {
        require(_affiliateFee < 100, "incorrect affiliate fee");
        affiliateFeeOnCreation = _affiliateFee;
    }
function setPlatformFee(uint _platfomFee) external onlyOwner {
        require(_platfomFee < 25, "incorrect platfom fee");
        platformFee = _platfomFee;
    }
function setCreationFee(uint _creationFee) external onlyOwner {
        creationFee = _creationFee;
    }
```

Mitigation:
Emit an event for critical parameter changes.

LOW RISK FINDING

Category: Centralization
Subject: Missing Zero Address
Status: Open
Severity: Low
Impact:

Functions can take a zero address as a parameter (0x00000...). If a function parameter of address type is not properly validated by checking for zero addresses, there could be serious consequences for the contract's functionality.

```
 function setPlatformAddress(address _feeAddress) external onlyOwner {
        feeAddress = _feeAddress;
    }
function setUpkeep(address _upkeep, address _forwarder) external onlyOwner {
        upkeep = ILogAutomation(_upkeep);
        forwarder = _forwarder;
    }
```

Mitigation:
Check that the address is not zero.

INFORMATIONAL RISK FINDING

Category: Optimization
Subject: Floating Pragma
Status: Open
Severity: Informational
Impact:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.4
```

Mitigation:
It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

INFORMATIONAL RISK FINDING

Category: Optimization
Subject: uint256
Status: Open
Severity: Informational
Impact:

Use uit256 instead of uint. uint is an alias for uint256 and is not recommended for use. The variable size should be clarified, as this can cause issues when encoding data with selectors if the alias is mistakenly used within the signature string.

```
uint public ticketPrice;
uint public maxTickets;
uint public endDate;
uint public creatorFee;
uint public charityFee;
uint public affiliateFee;
uint public startTime;
uint public maxTicketsPerWallet;
uint[] public prizeDistribution;
uint public platformFee;
```

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.