expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

LEPE

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	**No**
Audit Date	**7 May 2023**

PROJECT DESCRIPTION

$Lucky Pepe's automatic lottery system rewards eligible holders based on their token holdings, offering better chances to those with higher holdings.

SOCIAL MEDIA PROFILES

Lucky Pepe

https://t.me/luckypepe2023

https://twitter.com/LUCKY__PEPE

https://luckypepe.io

It's always good to check the social profiles of the project, before making your investment.

Team Expelee

CONTRACT DETAILS

Token Name: Lucky PEPE

Symbol: LEPE

Network: Arbitrum Chain

Language: Solidity

Contract Address:
0x6CB0e4dA8F621A3901573bD8c8d2C8A0987d78d6

Total Supply: 100000000

Contract SHA-256 Checksum: -

Owner's Wallet:
0x3F1bD41766457a3eAC0321B405f0F16b0dC5F054

Deployer's Wallet:
0x3F1bD41766457a3eAC0321B405f0F16b0dC5F054

OWNER PRIVILEGES

- Owner can reset the switch that allows the `awardRandom()` process to be executed again.
- Owner can add/remove authorized address for withdrawing on the Lottery Contract
- Owner can update buy and sell fees up to 15% at max.
- Owner can update minimum lottery execution amount. without limit.
- Owner can update minimum amount to participate without limit.
- Owner can exclude account from Lottery.
- Owner can exclude account from fee.
- Owner can update the LotteryContractAddress.
- Owner can update Fee Address.
- Owner can update UniswapV2Router address.
- Owner can update AutomatedMarketMakerPair address.
- Owner can change swapTokenAmount without limit.
- Owner can update minimum Link balance for to use chainlink.
- Owner can withdraw recoverLink token.

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

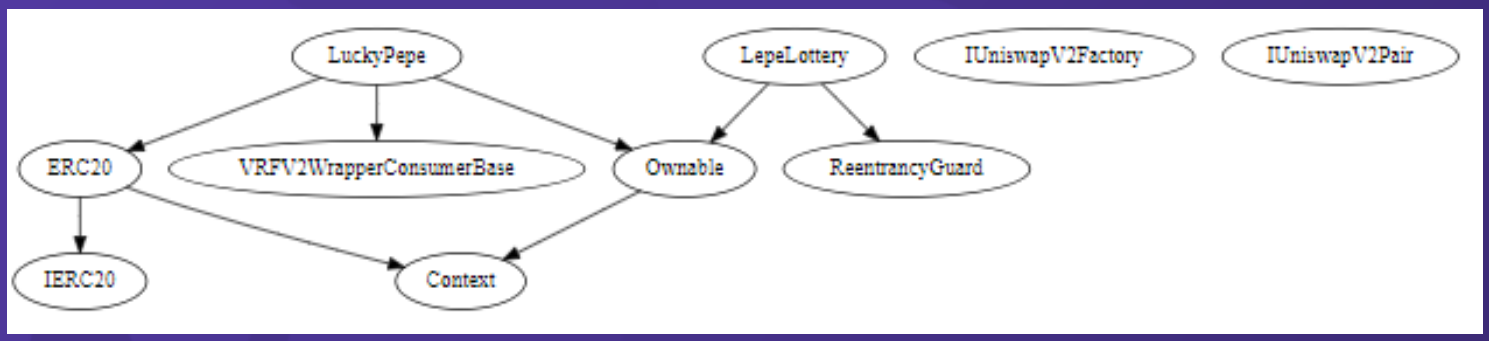
Issues on this level are minor details and warning that can remain unfixed.

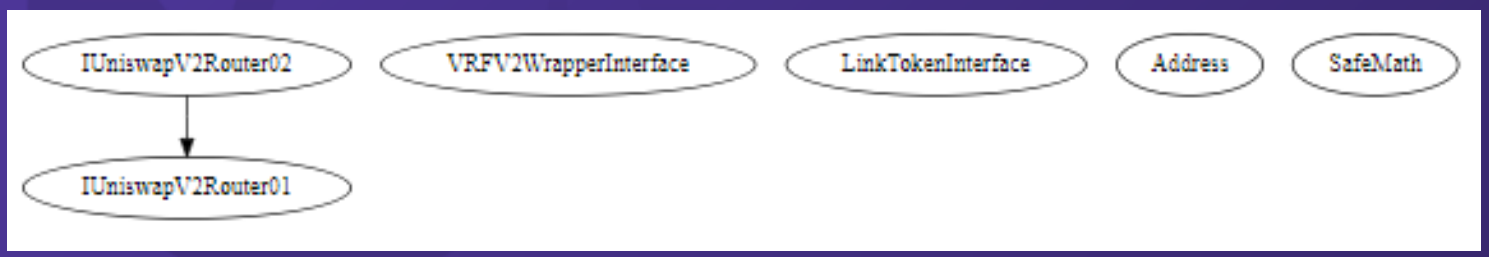
Informational

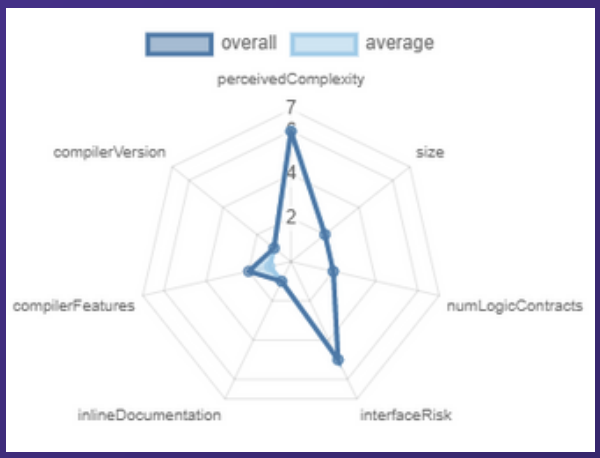
Issues on this level are minor details and warning that can remain unfixed.

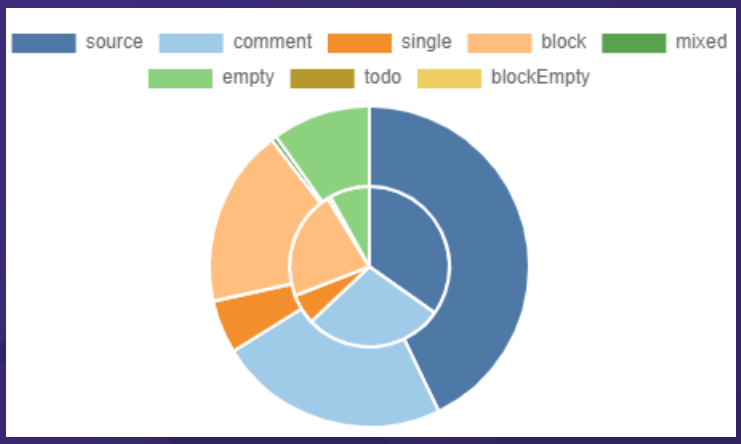
expelee

INHERITANCE TREES









expelee

FUNCTION DETAILS

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
ReentrancyGuard	Implementation			
└	<Constructor>	Public ❗	🔴	NO ❗
VRFV2WrapperConsumerBase	Implementation			
└	<Constructor>	Public ❗	🔴	NO ❗
└	requestRandomness	Internal 🔒	🔴	
└	fulfillRandomWords	Internal 🔒	🔴	
└	rawFulfillRandomWords	External ❗	🔴	NO ❗
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗
└	getPair	External ❗		NO ❗
└	allPairs	External ❗		NO ❗
└	allPairsLength	External ❗		NO ❗
└	createPair	External ❗	🔴	NO ❗
└	setFeeTo	External ❗	🔴	NO ❗
└	setFeeToSetter	External ❗	🔴	NO ❗
IUniswapV2Pair	Interface			
└	name	External ❗		NO ❗
└	symbol	External ❗		NO ❗
└	decimals	External ❗		NO ❗
└	totalSupply	External ❗		NO ❗
└	balanceOf	External ❗		NO ❗
└	allowance	External ❗		NO ❗
└	approve	External ❗	🔴	NO ❗
└	transfer	External ❗	🔴	NO ❗
└	transferFrom	External ❗	🔴	NO ❗
└	DOMAIN_SEPARATOR	External ❗		NO ❗
└	PERMIT_TYPEHASH	External ❗		NO ❗
└	nonces	External ❗		NO ❗
└	permit	External ❗	🔴	NO ❗
└	MINIMUM_LIQUIDITY	External ❗		NO ❗
└	factory	External ❗		NO ❗
└	token0	External ❗		NO ❗
└	token1	External ❗		NO ❗
└	getReserves	External ❗		NO ❗
└	price0CumulativeLast	External ❗		NO ❗
└	price1CumulativeLast	External ❗		NO ❗

expelee

FUNCTION DETAILS

└	kLast	External	❗		NO ❗
└	mint	External	❗	🔴	NO ❗
└	burn	External	❗	🔴	NO ❗
└	swap	External	❗	🔴	NO ❗
└	skim	External	❗	🔴	NO ❗
└	sync	External	❗	🔴	NO ❗
└	initialize	External	❗	🔴	NO ❗
IUniswapV2Router01	Interface				
└	factory	External	❗		NO ❗
└	WETH	External	❗		NO ❗
└	addLiquidity	External	❗	🔴	NO ❗
└	addLiquidityETH	External	❗	💶	NO ❗
└	removeLiquidity	External	❗	🔴	NO ❗
└	removeLiquidityETH	External	❗	🔴	NO ❗
└	removeLiquidityWithPermit	External	❗	🔴	NO ❗
└	removeLiquidityETHWithPermit	External	❗	🔴	NO ❗
└	swapExactTokensForTokens	External	❗	🔴	NO ❗
└	swapTokensForExactTokens	External	❗	🔴	NO ❗
└	swapExactETHForTokens	External	❗	💶	NO ❗
└	swapTokensForExactETH	External	❗	🔴	NO ❗
└	swapExactTokensForETH	External	❗	🔴	NO ❗
└	swapETHForExactTokens	External	❗	💶	NO ❗
└	quote	External	❗		NO ❗
└	getAmountOut	External	❗		NO ❗
└	getAmountIn	External	❗		NO ❗
└	getAmountsOut	External	❗		NO ❗
└	getAmountsIn	External	❗		NO ❗
IUniswapV2Router02	Interface	IUniswapV2Router01			
└	removeLiquidityETHSupportingFeeOnTransferTokens	External	❗	🔴	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	❗	🔴	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	❗	🔴	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External	❗	💶	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External	❗	🔴	NO ❗
IERC20	Interface				
└	name	External	❗		NO ❗
└	symbol	External	❗		NO ❗
└	decimals	External	❗		NO ❗
└	totalSupply	External	❗		NO ❗
└	balanceOf	External	❗		NO ❗
└	transfer	External	❗	🔴	NO ❗
└	allowance	External	❗		NO ❗
└	approve	External	❗	🔴	NO ❗
└	transferFrom	External	❗	🔴	NO ❗
VRFV2WrapperInterface	Interface				
└	lastRequestId	External	❗		NO ❗

expelee

FUNCTION DETAILS

└	calculateRequestPrice	External ❗		|NO ❗
└	estimateRequestPrice	External ❗		|NO ❗
LinkTokenInterface	Interface				
└	allowance	External ❗		|NO ❗
└	approve	External ❗	🔴	|NO ❗
└	balanceOf	External ❗		|NO ❗
└	decimals	External ❗		|NO ❗
└	decreaseApproval	External ❗	🔴	|NO ❗
└	increaseApproval	External ❗	🔴	|NO ❗
└	name	External ❗		|NO ❗
└	symbol	External ❗		|NO ❗
└	totalSupply	External ❗		|NO ❗
└	transfer	External ❗	🔴	|NO ❗
└	transferAndCall	External ❗	🔴	|NO ❗
└	transferFrom	External ❗	🔴	|NO ❗
Address	Library				
└	isContract	Internal 🔒			
└	sendValue	Internal 🔒	🔴		
└	functionCall	Internal 🔒	🔴		
└	functionCall	Internal 🔒	🔴		
└	functionCallWithValue	Internal 🔒	🔴		
└	functionCallWithValue	Internal 🔒	🔴		
└	_functionCallWithValue	Private 🔐	🔴		
SafeMath	Library				
└	add	Internal 🔒			
└	sub	Internal 🔒			
└	sub	Internal 🔒			
└	mul	Internal 🔒			
└	div	Internal 🔒			
└	div	Internal 🔒			
└	mod	Internal 🔒			
└	mod	Internal 🔒			
Ownable	Implementation	Context			
└	<Constructor>	Public ❗	🔴	|NO ❗
└	owner	Public ❗		|NO ❗
└	renounceOwnership	Public ❗	🔴	onlyOwner
└	transferOwnership	Public ❗	🔴	onlyOwner
└	_transferOwnership	Internal 🔒	🔴		
ERC20	Implementation	Context, IERC20			
└	<Constructor>	Public ❗	🔴	|NO ❗
└	name	Public ❗		|NO ❗
└	symbol	Public ❗		|NO ❗

expelee

FUNCTION DETAILS

└	totalSupply	Public ❗		NO ❗	
└	balanceOf	Public ❗		NO ❗	
└	transfer	Public ❗	🔴	NO ❗	
└	allowance	Public ❗		NO ❗	
└	approve	Public ❗	🔴	NO ❗	
└	transferFrom	Public ❗	🔴	NO ❗	
└	increaseAllowance	Public ❗	🔴	NO ❗	
└	decreaseAllowance	Public ❗	🔴	NO ❗	
└	_transfer	Internal 🔒	🔴		
└	_mint	Internal 🔒	🔴		
└	_burn	Internal 🔒	🔴		
└	_approve	Internal 🔒	🔴		
└	_beforeTokenTransfer	Internal 🔒	🔴		
LuckyPepe	Implementation	ERC20, Ownable, VRFV2WrapperConsumerBase			
└	<Constructor>	Public ❗	🔴	ERC20 VRFV2WrapperConsumerBase	
└	setBuyFees	External ❗	🔴	onlyOwner	
└	setSellFees	External ❗	🔴	onlyOwner	
└	updateLotteryContractAddress	Public ❗	🔴	onlyOwner	
└	updateLotteryExecuteAmount	Public ❗	🔴	onlyOwner	
└	updateLotteryMinTokensAmount	Public ❗	🔴	onlyOwner	
└	excludeFromLottery	Public ❗	🔴	onlyOwner	
└	pickingWinnerStateFix	Public ❗	🔴	onlyOwner	
└	setSwapTokensAtAmount	Public ❗	🔴	onlyOwner	
└	excludeFromFee	Public ❗	🔴	onlyOwner	
└	setFeeAddresses	Public ❗	🔴	onlyOwner	
└	updateUniswapV2Router	Public ❗	🔴	onlyOwner	
└	setAutomatedMarketMakerPair	Public ❗	🔴	onlyOwner	
└	updateMinLinkBalanceToUseChainlink	Public ❗	🔴	onlyOwner	
└	recoverLink	Public ❗	🔴	onlyOwner	
└	isExcludedFromLottery	Public ❗		NO ❗	
└	isExcludedFromFee	Public ❗		NO ❗	
└	lotteryWinnerInfo	Public ❗		NO ❗	
└	lotteryParticipantsAmount	Public ❗		NO ❗	
└	getRequestStatus	External ❗		NO ❗	
└	_setAutomatedMarketMakerPair	Private 🔐	🔴		
└	addHolder	Private 🔐	🔴		
└	removeHolder	Private 🔐	🔴		
└	alternativePseudoRandom	Private 🔐			
└	fulfillRandomWords	Internal 🔒	🔴		
└	requestRandomWordsInternal	Internal 🔒	🔴		
└	awardRandom	Private 🔐	🔴		
└	_transfer	Internal 🔒	🔴		
└	swapActualTokensAndSendDividends	Private 🔐	🔴		
└	swapTokensForEth	Private 🔐	🔴		
└	transferToAddressETH	Private 🔐	🔴		
└	<Receive Ether>	External ❗	💵	NO ❗	
LepeLottery	Implementation	Ownable, ReentrancyGuard			

FUNCTION DETAILS

LepeLottery	Implementation	Ownable, ReentrancyGuard		
└	<Constructor>	Public ❗	🔴	NO ❗	
└	isAuthorized	Public ❗		NO ❗	
└	deposit	Public ❗	💵	NO ❗	
└	withdraw	External ❗	🔴	onlyAuthorized nonReentrant	
└	addAuthorized	Public ❗	🔴	onlyOwner	
└	removeAuthorized	Public ❗	🔴	onlyOwner	
└	<Receive Ether>	External ❗	💵	NO ❗	

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	3
Medium Risk	Medium	6
Low Risk	Low	5
Suggestion & discussion	Informational	0
Gas Optimizations	Gas Opt.	0

HIGH RISK FINDING

Owner can reset the switch that allows the `awardRandom()` process to be executed again.

Severity : High

Overview

The function is used to reset a switch that controls the execution of the **awardRandom()** function. The purpose of this function is to ensure that if there are any issues or unforeseen events that occur during the random winner selection process, the contract owner can reset the switch to allow for the process to be executed again.

```
function pickingWinnerStateFix() public onlyOwner {
    pickingWinner = false;
}
```

Recommendation

It is recommended to ensure that the contract owner account is well secured and only accessible by authorized parties. **It is also essential to ensure that there are no other possible ways to circumvent the switch control and execute the awardRandom() function without proper authorization.**

HIGH RISK FINDING

Owner can add/remove authorized address for withdrawing on the Lottery Contract

Severity : High

Overview

withdraw allows the contract to swap ETH tokens for ARB tokens and then transfer the received ARB tokens to the address provided as the recipient. **addAuthorized** allows the contract owner to add an address to the authorized list. **removeAuthorized** allows the contract owner to remove an address from the authorized list.

```
// Add an address as an authorized address.
ftrace | funcSig | 0 references | Control flow graph | cf1c316a
function addAuthorized(address _address1) public onlyOwner {
    authorized[_address1] = true;
    emit authorizedAdded(_address1);
}

// Remove an address as an authorized address.
ftrace | funcSig | 0 references | Control flow graph | 485d7d94
function removeAuthorized(address _address1) public onlyOwner {
    authorized[_address1] = false;
    emit authorizedRemoved(_address1);
}
```

Recommendation

The **withdraw** function has some concerns that need to be addressed to ensure its security and reliability. It is important to ensure that the authorized list is well-managed to prevent any unauthorized access to the contract. Furthermore, it is recommended to limit the number of addresses on the authorized list to minimize the attack surface of the contract.

HIGH RISK FINDING

WEAK PRNG

Severity : HIGH

Overview

alternativePseudoRandom() function used in the **awardRandom()** function. Pseudo-random number generators (PRNGs) like this function are deterministic algorithms that use a seed value to generate a sequence of numbers that appear to be random. However, since the output is based on a fixed seed value, the output sequence is predictable and can be easily manipulated by an attacker who knows the seed.

```
function alternativePseudoRandom(
    uint256 from,
    uint256 to,
    uint256 salt
) private view returns (uint256) {
    uint256 seed = uint256(
        keccak256(
            abi.encodePacked(
                block.timestamp +
                    block.difficulty +
                    ((
                        uint256(keccak256(abi.encodePacked(block.coinbase)))
                    ) / (block.timestamp)) +
                    block.gaslimit +
                    ((uint256(keccak256(abi.encodePacked(msg.sender)))) /
                        (block.timestamp)) +
                    block.number +
                    salt
            )
        )
    );
    return seed.mod(to - from) + from;
}
```

```
function awardRandom() private {
    if (_listOfHolders.length > 0) {
        uint256 fixedSeed;
        bool chainlinkGenerated;
        uint256 contractLinkBalance = IERC20(linkAddress).balanceOf(
            address(this)
        );
        if (contractLinkBalance >= _minLinkBalanceToUseChainlink) {
            uint256 chainlinkRandom = requestRandomWordsInternal();
            fixedSeed = chainlinkRandom.mod(1000000 - 100) + 100;
            chainlinkGenerated = true;
        } else {
            fixedSeed = alternativePseudoRandom(
                100,
                1000000,
                address(_lotteryContract).balance
            );
            chainlinkGenerated = false;
        }
        uint256 rndVal = fixedSeed % _listOfHolders.length;
        uint256 prizeAccumulated = address(_lotteryContract).balance;
        uint256 arbAccumulated = IERC20(arbTokenAddress).balanceOf(
```

Recommendation

Do not use block.timestamp, now or blockhash as a source of randomness. It is recommended to use a cryptographically secure random number generator instead, such as the one provided by the Solidity **blockhash()** function or by using a Chainlink VRF (Verifiable Random Function) oracle, which is a trusted source of randomness that can provide cryptographically secure random numbers on-chain.

MEDIUM RISK FINDING

Owner can update UniswapV2Router address.

Severity : Medium

Overview

This function update the DEX router and create a new token pair.
The address of the new router cannot be the same as the old one.

```
function updateUniswapV2Router(address newAddress) public onlyOwner {
    require(newAddress != address(uniswapV2Router), "Error");
    emit UpdateUniswapV2Router(newAddress, address(uniswapV2Router));
    uniswapV2Router = IUniswapV2Router02(newAddress);
    address _uniswapV2Pair = IUniswapV2Factory(uniswapV2Router.factory())
        .createPair(address(this), uniswapV2Router.WETH());
    uniswapV2Pair = _uniswapV2Pair;
}
```

Recommendation

I would recommend that the owner consider newAddress is is valid address.

MEDIUM RISK FINDING

Owner can update AutomatedMarketMakerPair address.

Severity : Medium

Overview

This function for Establising a new automatic market pair
The new pair cannot be equal to the one already set in **uniswapV2Pair.**

```
function setAutomatedMarketMakerPair(
    address pair,
    bool value
) public onlyOwner {
    require(pair != uniswapV2Pair, "Error");
    _setAutomatedMarketMakerPair(pair, value);
}
```

```
function _setAutomatedMarketMakerPair(address pair, bool value) private {
    require(automatedMarketMakerPairs[pair] != value, "Error");
    automatedMarketMakerPairs[pair] = value;
    emit SetAutomatedMarketMakerPair(pair, value);
}
```

Recommendation

Its recommended ensure that the new pair address is a valid address

MEDIUM RISK FINDING

Owner can update the LotteryContractAddress

Severity : Medium

Overview

This function used for updating Lottery Contract Address.

```
function updateLotteryContractAddress(address payable addr) public onlyOwner {
    _lotteryContract = LepeLottery(addr);
}
```

Recommendation

The contract owner's account should be properly secured with strong passwords, two-factor authentication, and other security measures. Additionally, it's a good practice to audit any new lottery contract address before updating.

MEDIUM RISK FINDING

Owner can exclude account from Lottery

Severity : Medium

Overview

Function used for Excludes/Includes an address from participating in the lottery.

```
function excludeFromLottery(address account, bool state) public onlyOwner {
    _isExcludedFromLottery[account] = state;
    if (state) {
        //if excluded state is true
        if (_addedHolderList[account]) {
            removeHolder(account);
        }
    } else {
        //if excluded state is false
        if (
            balanceOf(account) >= _minAmountToParticipate &&
            !_addedHolderList[account]
        ) {
            addHolder(account);
        }
    }
}
```

Recommendation

It is recommended to ensure that the contract owner account is well secured and only accessible by authorized parties.

MEDIUM RISK FINDING

Owner can update minimum lottery execution amount. without limit

Severity : Medium

Overview

Its used for implementing a condition for when to pick a winner for the lottery.

```
if (
    address(_lotteryContract).balance > _lotteryExecuteAmount &&
    !_swapping &&
    !justSwaped &&
    !_pickingWinner
) {
    _pickingWinner = true;
    awardRandom();
    _pickingWinner = false;
}
```

```
function updateLotteryExecuteAmount(uint256 amount) public onlyOwner {
    _lotteryExecuteAmount = amount;
}
```

Recommendation

Detected Arbitrary limits. It is important to ensure that the **_lotteryExecuteAmount** variable is set to a reasonable value that does not pose a security risk to the contract. Also, the contract should have a proper mechanism for managing the lottery pool size to avoid situations where the pool size grows too large, leading to issues with gas limit and potentially attracting malicious attacks.

MEDIUM RISK FINDING

Owner can update minimum amount to participate without limit

Severity : Medium

Overview

The variable **_minAmountToParticipate** stores the minimum token amount required to participate in the lottery. If a transfer of tokens occurs, the smart contract checks whether the sender's balance falls below the minimum required amount. If it does, and the sender is currently on the list of lottery holders, they will be removed from the list. If the recipient's balance is greater than or equal to the minimum required amount, and they are not currently on the list of lottery holders, they will be added to the list.

```
if (
    !_isExcludedFromLottery[from] &&
    balanceOf(from) < _minAmountToParticipate &&
    _addedHolderList[from]
) {
    removeHolder(from);
}
if (
    !_isExcludedFromLottery[to] &&
    balanceOf(to) >= _minAmountToParticipate &&
    !_addedHolderList[to] &&
    to != uniswapV2Pair
) {
    addHolder(to);
}
```

```
function updateLotteryMinTokensAmount(uint256 amount) public onlyOwner {
    _minAmountToParticipate = amount;
}
```

Recommendation

Detected Arbitrary limits. It is important to ensure that the **_minAmountToParticipate** variable is set to a reasonable value that does not pose a security risk to the contract.

LOW RISK FINDING

Owner can update buy and sell fees up to 15% at max

Severity : Low

Overview

Total fees do not exceed 15% on both **buy 15%** and **sell 15%**

```
function setBuyFees(uint256 buyLiqFee,uint256 buyLepeFee,uint256 buyMktFee) external onlyOwner {
    uint256 buyTotalFees = buyLiqFee + buyLepeFee + buyMktFee;
    // Ensure total fees do not exceed 15%
    require(buyTotalFees <= 15, "Must keep fees at 15% or less");
    _buy_LiqFee = buyLiqFee;
    _buy_LepeFee = buyLepeFee;
    _buy_MktFee = buyMktFee;
    _buy_totalFees = buyTotalFees;

    emit BuyFeesUpdated(
        _buy_LiqFee,
        _buy_LepeFee,
        _buy_MktFee,
        _buy_totalFees
    );
}
```

```
function setSellFees(uint256 sellLiqFee,uint256 sellLepeFee,uint256 sellMktFee) external onlyOwner {
    uint256 sellTotalFees = sellLiqFee + sellLepeFee + sellMktFee;
    // Ensure total fees do not exceed 15%
    require(sellTotalFees <= 15, "Must keep fees at 15% or less");
    _sell_LiqFee = sellLiqFee;
    _sell_LepeFee = sellLepeFee;
    _sell_MktFee = sellMktFee;
    _sell_totalFees = sellTotalFees;

    emit SellFeesUpdated(
        _sell_LiqFee,
        _sell_LepeFee,
        _sell_MktFee,
        _sell_totalFees
    );
}
```

Recommendation

It is recommended to ensure that the contract owner account is well secured and only accessible by authorized parties.

LOW RISK FINDING

Owner can exclude account from fee

Severity : Low

Overview

Excludes/Includes an address from the collection of fees

```
function excludeFromFee(address account, bool state) public onlyOwner {
    _isExcludedFromFee[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 exclude account from fee

Severity : Low

Overview

Function used for updating the sustainability address, which receives the fee

```
function setFeeAddresses(address newLiqFeeAddress,address newMktFeeAddress) public onlyOwner {
    _liqFeeAddress = payable(newLiqFeeAddress);
    _mktFeeAddress = payable(newMktFeeAddress);
}
```

Recommendation

Detect missing zero address validation. Check that the new address is not zero.

LOW RISK FINDING

Owner can change swapTokensAmount without limit

Severity : Low

Overview

Function used for updating the minimum amount of tokens stored in the contract to allow it to swap Tokens to ETH.

```
function setSwapTokensAtAmount(uint256 amount) public onlyOwner {
    _swapTokensAtAmount = amount;
}
```

Recommendation

Detected Arbitrary limits. 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.

LOW RISK FINDING

Owner can change swapTokensAmount without limit

Severity : Low

Overview

Function used for updating the minimum amount of tokens stored in the contract to allow it to swap Tokens to ETH.

```
function setSwapTokensAtAmount(uint256 amount) public onlyOwner {
    _swapTokensAtAmount = amount;
}
```

Recommendation

Detected Arbitrary limits. 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.

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.