# enpelee

Building the Futuristic Blockchain Ecosystem

## Audit Report FOR







## OVERVIEW

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	Not Done
Audit Date	2 Sep 2022

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

- Team Expelee



## PROJECT DESCRIPTION

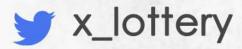
## Lottery-X Token

Filled by altruistic principle, aiming to give back to the community and beyond.

We are bringing the lottery onto the blockchain ecosystem. Launched on BSC on the back of an idea of two ambitious crypto investors with great track record, Lottery-X offers a very simple concept







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

- Team Expelee





## **CONTRACT DETAILS**

Token Name

Lottery-X

Symbol

**SRYX** 

Contract Address

0x237aF566785EA1a717363EDA306820140641f78a

Network

BSC

Language

Solidity

Total Supply

13,000,000,000

**Decimals** 

18

Compiler

v0.8.16+commit.07a7930e

License

default license



## **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
- Complier
- Hardhat





## **FUNCTION OVERVIEW**

Can Take Back Ownership

**Owner Change Balance** 

**Blacklist** 

**Modify Fees** 

**Proxy** 

**Whitelisted** 

**Anti Whale** 

**Trading Cooldown** 

**Transfer Pausable** 

Cannot Sell All

**Hidden Owner** 

**Creator Address** 

**Creator Balance** 

**Owner Address** 

Mint

**Detected** 

**Not Detected** 

**Detected** 

**Detected** 

**Not Detected** 

**Not Detected** 

**Detected** 

**Not Detected** 

**Not Detected** 

**Not Detected** 

**Not Detected** 

0x559db94efc5debf54a1b1c35ed881d1e96a67141

13,000,000,000 RYX

0x559db94efc5debf54a1b1c35ed881d1e96a67141

**Not Detected** 





## **VULNERABILITY CHECKLIST**

Design Logic	Passed
Compiler warnings.	Passed
Private user data leaks	Passed
Timestamp 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 Zeppelin module	Passed
Fallback function security	Passed



## RISK CLASSIFICATION

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access 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

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.



## **MANUAL AUDIT**

Lottery-X is a RFI token

#### Contract SHA256 Checksum:

a2f7a242a4fe5aa69a6e5ded4e42356cde9f039108993b3fc3bb8cfa23e3a30f

#### **Centralization Risks:**

#### High:

Owner is able to set taxes up to 50% on buy and 50% on sell

#### High:

Owner is able to blacklist an arbitrary address from transfering, buying and selling tokens

```
function blacklistAddress(address account, bool value) external onlyOwner {
    _isBlacklisted[account] = value;
}
```

#### Medium:

Owner is able to set a limit for maximum amount of tokens that can be traded or be holded in wallet, this limit can not be less than 1% of total supply

#### Medium:

Owner can take ownership of contrat back after locktime using unlock function, owner of contract is able to change fees, blacklist users and include/exclude wallets from fees.



#### Logical Issues

#### Critical:

```
contractTokenBalance >= numTokensSellToAddToLiquidity
```

there is no function to change this rewardToken

#### **Recommendation:**

set **rewardToken** to a valid ERC20 token (eg: doge, busd, usdt, etc) at time of initialization and also set a function to be able to change it later

#### High:

setting contractTokenBalance to \_maxTxAmount can disable all contract inner balance (AKA collected fees from trades) swaps if

```
_maxTxAmount < numTokensSellToAddToLiquidity

its not clear what is the usage of this if statement:

if(contractTokenBalance >= _maxTxAmount){
```

contractTokenBalance = \_maxTxAmount;

#### **Recommendation:**

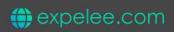
remove this if statement from code

#### High:

}

name of function tells that we are going to calculate only liquidity fee, but its using all other fees as well.





#### **Recommendation:**

this might be intentional, if it is, change name of function to something like "calculateTaxes" if not, delete other fees

#### Medium:

contractTokenBalance = numTokensSellToAddToLiquidity;

setting contractTokenBalance to numTokensSellToAddToLiquidity even if:

contractTokenBalance >= numTokensSellToAddToLiquidity

can cause some tokens to be stuck in contract, specially when there is huge amount of trades

contractTokenBalance may be way bigger than numTokensSellToAdd
ToLiquidity

#### Recommendation:

remove

contractTokenBalance = numTokensSellToAddToLiquidity;

#### **Suggestions:**

- spentAmount = contractTokenBalance.div(totFee).mul(\_burnFee);
   do multiply prior to divide
- since we are already giving reflections to holders, it might not be necessary to use a dividend tracker
- Potential Error at updatePcsV2Router due to ABI difference:
   new router may not follow the ABI of Uniswap for creating pair, this is pretty
   much unlikely, but to make sure that there wont be any problem in future, try
   sending calldata of creating pair operation as an input parameter for new
   router address and then create pair with low level call to new router.
- emit an event in this functions:
   updatePcsV2Router excludeFromReward includeInReward setAllFeePercent
   setBuybackUpperLimit | no limit setMaxTxPercent setMaxWalletPercent
   setFeeWallet setFeeWalletCharity setWalletFeeTokenType
   setWalletCharityFeeTokenType
   setMinimumTokenBalanceForDividends

expelee.com



#### **Gas Optimizations:**

#### Medium:

removeAllFee reads and write a from and to storage, instead of removing fees before transfers for excluded wallets, use a boolean like takeFee = false, to avoid taking fees and also avoid huge amount of gas that removeAllFee uses, this is the same for restoreAllFee, you can avoid this huge amounts of gas usage buy only using a boolean like takeFee

#### **Recommendation:**

```
if (ExcludedFromFees[from] || ExcludedFromFees[to]){
   takeFee = false;
}else {
   takeFee = true;
}
```

#### Medium

there is a massive gas usage while reading and writing to storage because of this bad practices; instead of this redundant reads from storage

```
_liquidityFee + _burnFee + _walletFee + _buybackFee + _walletCharityFee + _rewardFee4

if(_taxFee == 0 && _liquidityFee == 0 && _burnFee == 0 && _walletFee == 0 && _buybackFee == 0 && _walletCharityFee == 0 && _rewardFee == 0) return;

save this all of fees in a variable called "totalFees" and then check related conditions using that variable
```

#### Other (Low):

- Line 849 => define dead as constant to save gas
- Line 902 => define router as immutable
- Line 938 => define pcsV2Router as immutable
- Lines 1041 1049 => redundant check for fees, its checking wether they are equl or greater than zero and because this fees are of type uint8 they are always equal to or greater than 0
- move line 1409 to top of other require statements & remove lines 1401 & 1047
- Line 1050 and 1056 are redundant as this fees must be 0 at time of initializing contract
- no need to use SafeMath Libraries as they are compilers > 0.8.0 has a safemath internally, using SafeMath only increases gas with no point





#### Define this functions as external:

updatePcsV2Router excludeFromReward includeInReward setSwapAndLiquifyEnabled external: updatePcsV2Router excludeFromReward includeInReward setSwapAndLiquifyEnabled

#### external:

updatePcsV2Router excludeFromReward includeInReward setSwapAndLiquifyEnabled



### **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

- y expeleeofficial
- expelee

Expelee

- m expelee
- o expelee\_official
- 👩 expelee-co



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 guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect 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. Always 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.