expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Liverpool Fans Coin

TOKEN OVERVIEW

Risk Findings

Severity	Found
🔴 High	0
🟠 Medium	0
🟡 Low	2
🔵 Informational	0

Centralization Risks

Owner Privileges	Description
🟢 Can Owner Set Taxes >25% ?	Not Detected
🟢 Owner needs to enable trading ?	Not Detected
🟢 Can Owner Disable Trades ?	Not Detected
🟢 Can Owner Mint ?	Not Detected
🟢 Can Owner Blacklist ?	Not Detected
🟢 Can Owner set Max Wallet amount ?	Not Detected
🟢 Can Owner Set Max TX amount ?	Not Detected

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	**21 November 2023**

CONTRACT DETAILS

Token Address:
0x234972250880E8631FF6cbe3f92E286b40365985

Name: Liverpool Fans Coin

Symbol: LFC

Decimals: 18

Network: Binance smart chain

Token Type: ERC20

Owner: 0xf59EE4d75E23E60b3Aa6Fb35cfe02eA6dF146E1E

Deployer: 0xf59EE4d75E23E60b3Aa6Fb35cfe02eA6dF146E1E

Token Supply: 21000000000000000000000000

Checksum: 37265763766ad32e37ad6b85aad793e9

Testnet version:
The tests were performed using the contract deployed on the Binance smart chain Testnet, which can be found at the following address:
https://testnet.bscscan.com/address/0x0c1250da6c69ce88fdf2946bd0de64339378bdba#code

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.

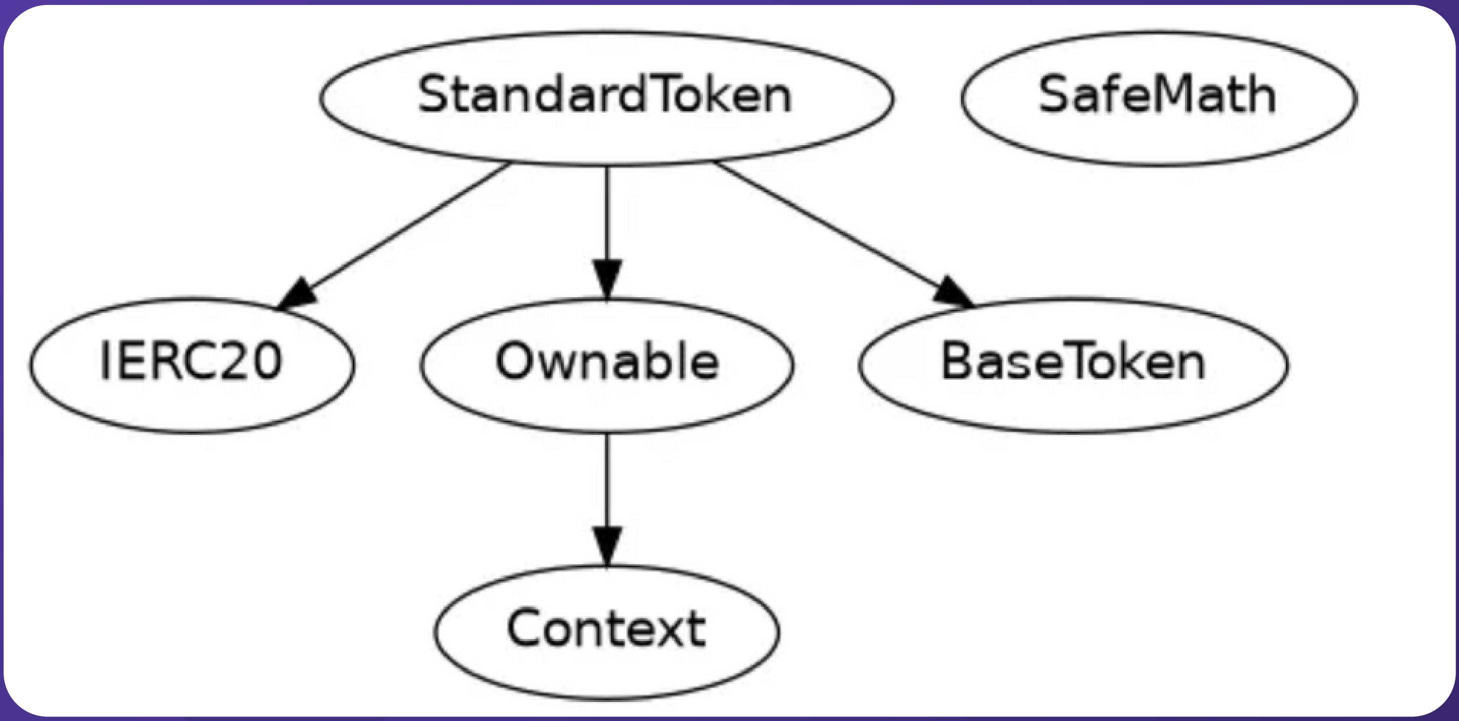
Low Risk

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

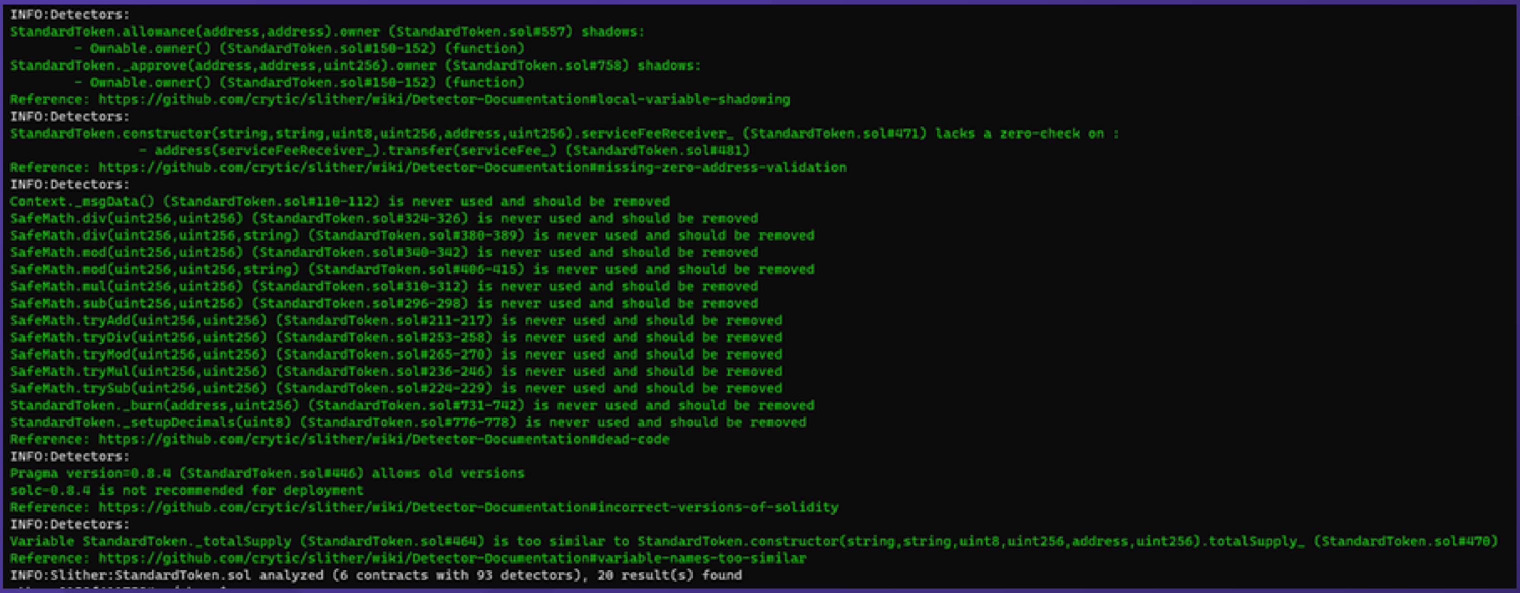
Informational

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

INHERITANCE TREES



STATIC ANALYSIS



```
INFO:Detectors:
StandardToken.allowance(address,address).owner (StandardToken.sol#557) shadows:
        - Ownable.owner() (StandardToken.sol#150-152) (function)
StandardToken._approve(address,address,uint256).owner (StandardToken.sol#758) shadows:
        - Ownable.owner() (StandardToken.sol#150-152) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
StandardToken.constructor(string,string,uint8,uint256,address,uint256).serviceFeeReceiver_ (StandardToken.sol#471) lacks a zero-check on :
                - address(serviceFeeReceiver_).transfer(serviceFee_) (StandardToken.sol#481)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Context._msgData() (StandardToken.sol#110-112) is never used and should be removed
SafeMath.div(uint256,uint256) (StandardToken.sol#324-326) is never used and should be removed
SafeMath.div(uint256,uint256,string) (StandardToken.sol#380-389) is never used and should be removed
SafeMath.mod(uint256,uint256) (StandardToken.sol#340-342) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (StandardToken.sol#406-415) is never used and should be removed
SafeMath.mul(uint256,uint256) (StandardToken.sol#310-312) is never used and should be removed
SafeMath.sub(uint256,uint256) (StandardToken.sol#296-298) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (StandardToken.sol#211-217) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (StandardToken.sol#253-258) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (StandardToken.sol#265-270) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (StandardToken.sol#236-246) is never used and should be removed
SafeMath.trySub(uint256,uint256) (StandardToken.sol#224-229) is never used and should be removed
StandardToken._burn(address,uint256) (StandardToken.sol#731-742) is never used and should be removed
StandardToken._setupDecimals(uint8) (StandardToken.sol#776-778) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.4 (StandardToken.sol#446) allows old versions
solc-0.8.4 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable StandardToken._totalSupply (StandardToken.sol#464) is too similar to StandardToken.constructor(string,string,uint8,uint256,address,uint256).totalSupply_ (StandardToken.sol#470)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Slither:StandardToken.sol analyzed (6 contracts with 93 detectors), 20 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0xe0906e0e9b37575505ef092e3dd89d9dfd7b14
3ff3ce8a0894055feec8196277

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x3e5d33dc58239957d79df3116f09d712b2035cf
ac3528ce30c43b0e34eabac5f

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x0b8c0369d859b6506f07168cbe2f099cadef83
afcbb57ea27977f709caf4cd9f

4- Transfer (passed):
https://testnet.bscscan.com/tx/0xf4bc083d943ecee972a7ff4e2164b0ea37b9bec
9dd1e32935eda5bcef985d036

5- Renounce Ownership (passed):
https://testnet.bscscan.com/tx/0xffb50953dd04125ad07dd0bd563693787dee15
eaaa163ec57ec8d2472812e9ac

MANUAL REVIEW

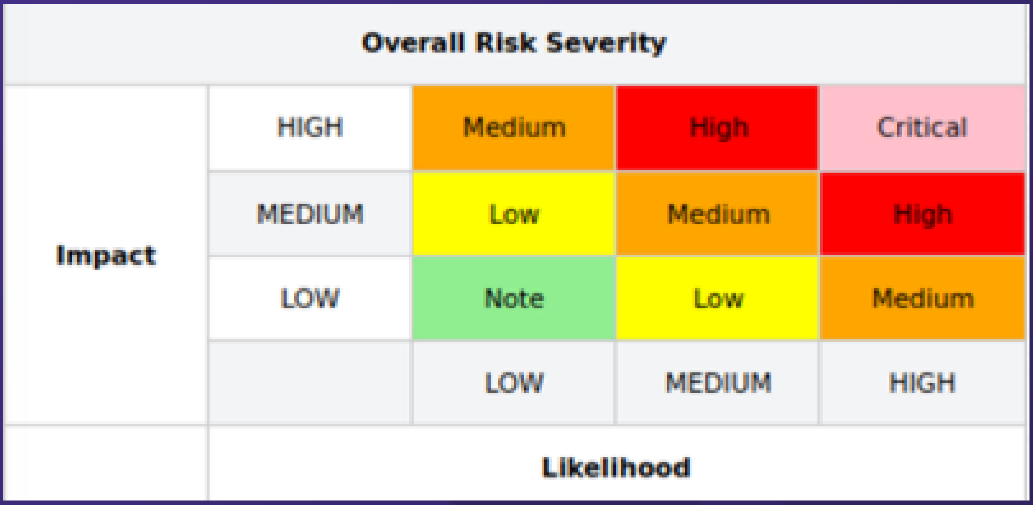
Severity Criteria

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

Centralization – Missing Zero Address

Severity: Low

Status: Open

Overview:
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.

```
constructor(
    string memory name_,
    string memory symbol_,
    uint8 decimals_,
    uint256 totalSupply_,
    address serviceFeeReceiver_,
    uint256 serviceFee_
  ) payable {
    _name = name_;
    _symbol = symbol_;
    _decimals = decimals_;
    _mint(owner(), totalSupply_);

    emit TokenCreated(owner(), address(this), TokenType.standard,
VERSION);

    payable(serviceFeeReceiver_).transfer(serviceFee_);
  }
```

Suggestion:
It is suggested that the address should not be zero or dead.

LOW RISK FINDING

Centralization – Remove the safe math library.
Severity: Low
Status: Open
Line Number: 205-416

Overview:
The Safe Math library is no longer needed for Solidity version 0.8 and above. This is because Solidity 0.8 includes checked arithmetic operations by default. All of Safe Math’s methods are now inherited into Solidity programming.

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_official
expelee
expelee
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.

Building the Futuristic Blockchain Ecosystem