



Building the Futuristic **Blockchain Ecosystem**

# SECURITY AUDIT REPORT

FT

# 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

02	Token Overview	
03	Table of Contents	
04	Overview	
05	Contract Details	
06	Audit Methodology	
07	Vulnerabilities Checklist	
08	Risk Classification	
09	Inheritance Trees	
10	Function Details	
13	Manual Review	
14	Low Risk Finding	
17	About Expelee	
18	Disclaimer	

# 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:

<b>Audit Result</b>	<b>Passed</b>
<b>KYC Verification</b>	-
<b>Audit Date</b>	<b>22 July 2023</b>

# CONTRACT DETAILS

Token Name: FT Token

Symbol: FT

Network: Binance Smart Chain

Language: Solidity

Contract Address:

0x0254b8eE29F79c042A4ac77E1CAA079f5f656955

Total Supply: 10,000,000

Owner's Wallet:

0xB1E24b35f125d3304B4e0D9ED428ddc82B072514

Deployer's Wallet:

0xd9B535BF39d9EefeeA15ef46C378D04cd7c7e719

# 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 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

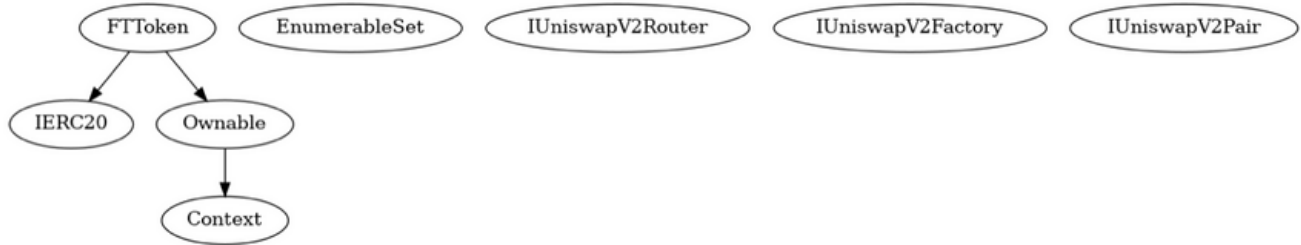
Issues on this level are minor details and warnings that can remain unfixed.

## Informational

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



# INHERITANCE TREES



# FUNCTION DETAILS

```

| Contract |      Type      |      Bases      |      |      |
|:-----:|:-----:|:-----:|:-----:|:-----:|
|  L  | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
|||||
| **Context** | Implementation | |||
|  L  | _msgSender | Internal  | ||
|  L  | _msgData | Internal  | ||
|||||
| **IERC20** | Interface | |||
|  L  | totalSupply | External  | | NO  |
|  L  | balanceOf | External  | | NO  |
|  L  | transfer | External  |  | NO  |
|  L  | allowance | External  | | NO  |
|  L  | approve | External  |  | NO  |
|  L  | transferFrom | External  |  | NO  |
|||||
| **Ownable** | Implementation | Context |||
|  L  | <Constructor> | Public  |  | NO  |
|  L  | owner | Public  | | NO  |
|  L  | _checkOwner | Internal  | ||
|  L  | renounceOwnership | Public  |  | onlyOwner |
|  L  | transferOwnership | Public  |  | onlyOwner |
|  L  | _transferOwnership | Internal  |  ||
|||||
| **EnumerableSet** | Library | |||
|  L  | _add | Private  |  ||
|  L  | _remove | Private  |  ||
|  L  | _contains | Private  | ||
|  L  | _length | Private  | ||

```

# FUNCTION DETAILS

```

|  ↳ | _at | Private 🔒 | || |
|  ↳ | _values | Private 🔒 | ||
|  ↳ | add | Internal 🔒 | 🔴 | ||
|  ↳ | remove | Internal 🔒 | 🔴 | ||
|  ↳ | contains | Internal 🔒 | ||
|  ↳ | length | Internal 🔒 | ||
|  ↳ | at | Internal 🔒 | ||
|  ↳ | values | Internal 🔒 | ||
|  ↳ | add | Internal 🔒 | 🔴 | ||
|  ↳ | remove | Internal 🔒 | 🔴 | ||
|  ↳ | contains | Internal 🔒 | ||
|  ↳ | length | Internal 🔒 | ||
|  ↳ | at | Internal 🔒 | ||
|  ↳ | values | Internal 🔒 | ||
|  ↳ | add | Internal 🔒 | 🔴 | ||
|  ↳ | remove | Internal 🔒 | 🔴 | ||
|  ↳ | contains | Internal 🔒 | ||
|  ↳ | length | Internal 🔒 | ||
|  ↳ | at | Internal 🔒 | ||
|  ↳ | values | Internal 🔒 | ||
|||||
| **IpancakeswapV2Router** | Interface | |||
|  ↳ | factory | External 🔴 | |NO 🔴 |
|||||
| **IpancakeswapV2Factory** | Interface | |||
|  ↳ | createPair | External 🔴 | 🔴 |NO 🔴 |
|||||
| **IpancakeswapV2Pair** | Interface | |||
|  ↳ | factory | External 🔴 | |NO 🔴 |
|  ↳ | token0 | External 🔴 | |NO 🔴 |
|  ↳ | token1 | External 🔴 | |NO 🔴 |
|  ↳ | totalSupply | External 🔴 | |NO 🔴 |
|  ↳ | balanceOf | External 🔴 | |NO 🔴 |
|  ↳ | getReserves | External 🔴 | |NO 🔴 |
|||||
| **FTToken** | Implementation | IERC20, Ownable |||
|  ↳ | <Constructor> | Public 🔴 | 🔴 |NO 🔴 |

```

# FUNCTION DETAILS

```

|  ↳ setLpAwardCondition | External  !  |  🔴  | onlyOwner |
|  ↳ getInteractionInfo | External  !  |  |NO !  |
|  ↳ addrIsInLpHolders | External  !  |  |NO !  |
|  ↳ setTradeOpen | External  !  |  🔴  | onlyOwner |
|  ↳ setInteraction | External  !  |  🔴  | onlyOwner |
|  ↳ checkSetLpShare | External  !  |  🔴  |NO !  |
|  ↳ name | Public  !  |  |NO !  |
|  ↳ symbol | Public  !  |  |NO !  |
|  ↳ decimals | Public  !  |  |NO !  |
|  ↳ totalSupply | Public  !  |  |NO !  |
|  ↳ balanceOf | Public  !  |  |NO !  |
|  ↳ transfer | Public  !  |  🔴  |NO !  |
|  ↳ allowance | Public  !  |  |NO !  |
|  ↳ approve | Public  !  |  🔴  |NO !  |
|  ↳ transferFrom | Public  !  |  🔴  |NO !  |
|  ↳ increaseAllowance | Public  !  |  🔴  |NO !  |
|  ↳ decreaseAllowance | Public  !  |  🔴  |NO !  |
|  ↳ _spendAllowance | Internal  🔒  |  🔴  | |
|  ↳ _approve | Internal  🔒  |  🔴  | |
|  ↳ _doDividend | Private  🔒  |  🔴  | |
|  ↳ _isLiquidity | Internal  🔒  |  | |
|  ↳ _handleTakeFee | Private  🔒  |  🔴  | |
|  ↳ _transfer | Private  🔒  |  🔴  | |
|  ↳ processLpDividend | Private  🔒  |  🔴  | |
|  ↳ setLpShare | Private  🔒  |  🔴  | |
|  ↳ checkLpAwardCondition | Internal  🔒  |  | |

```

## ### Legend

```

| Symbol | Meaning |
|:-----:|:-----|
|  🔴  | Function can modify state |
|  🔒  | Function is payable |

```

# MANUAL REVIEW

## Severity Criteria

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

Vulnerabilities are divided into three primary risk categories:

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

## Duplicate share holder

Category: **Logical**

Status: **Closed** – Contract is live and immutable

Impact: **Low**

### Overview:

Even if a share holder is already added, it will be added to lpHolders again if checkLpAwardCondition returns true.

```
function setLpShare(address owner) private {  
    if (lpHolders.contains(owner)) {  
        // User has Removed lp  
        if (!checkLpAwardCondition(owner)) {  
            lpHolders.remove(owner);  
        }  
        return;  
    }  
    // User is not in lpHolder  
    if (checkLpAwardCondition(owner)) {  
        lpHolders.add(owner);  
    }  
}
```

# LOW RISK FINDING

## Suggestion:

To mitigate this issue, its recommended to check whether the “owner” is already added to lpHolder or not. This could be achieved by adding an “else” before the second condition

```
function setLpShare(address owner) private {  
    if (lpHolders.contains(owner)) {  
        // User has Removed lp  
        if (!checkLpAwardCondition(owner)) {  
            lpHolders.remove(owner);  
        }  
        return;  
    }  
    // User is not in lpHolder  
    else if (checkLpAwardCondition(owner)) {  
        lpHolders.add(owner);  
    }  
}
```



# LOW RISK FINDING

## Misinterpretation of liquidity addition/removal

**Category:** Logical

**Status:** Closed – Contract is live and immutable

**Impact:** Low

### Overview:

\_isLiquidity function is trying to determine whether the transaction is adding or removing liquidity based on the USDT balance in the pancakeswap V2 pair contract. However, when adding or removing liquidity in pancakeswap V2, both tokens (tokenA and tokenB, or in this case your token and USDT) are transferred in the same transaction, but the transfers do not occur simultaneously.

For example, when adding liquidity, the function addLiquidity in the pancakeswap V2 Router contract will cause both tokens to be transferred to the pair contract. However, the ERC20 transferFrom function is called separately for each token. The order in which these transfers happen is important. If your token is token0 in the pair, it will be transferred first, followed by the USDT. If your token is token1, the USDT will be transferred first.

As a result, when \_isLiquidity is called during the transfer of your token, the USDT balance may not yet be updated, leading to potentially incorrect results.

### Suggestion:

To mitigate this issue, consider utilizing events emitted by the pancakeswap contracts to track liquidity changes, which can be monitored off-chain. Alternatively, redesign the logic for checking liquidity changes to account for the order of transfers in pancakeswap transactions.



# 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](http://www.expelee.com)

 [expeleeofficial](https://twitter.com/expeleeofficial)

 [expelee](https://medium.com/expelee)

 [Expelee](https://t.me/Expelee)

 [expelee](https://in.linkedin.com/company/expelee)

 [expelee\\_official](https://www.instagram.com/expelee_official)

 [expelee-co](https://github.com/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.

The logo for Expelee, featuring the word "expelee" in a stylized font. The "ex" is in white, and "pelee" is in orange. The letters are bold and modern.

Building the Futuristic **Blockchain Ecosystem**