expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

JESUS

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 with High Issue
KYC Verification	Not Done
Audit Date	28 May 2023

CONTRACT DETAILS

Token Name:

Symbol:

Network: Binance Smart Chain

Language: Solidity

Contract Address: local file

Total Supply: -

Owner's Wallet: local file

Deployer's Wallet: local file

Testnet Link:
https://testnet.bscscan.com/address/0x3399df5abE9F27bc73474a2E491685E26Cca741A

OWNER PRIVILEGES

- Reward calculation logic error
- Owner can set hold date max 10
- Owner can withdraw stuck tokens

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.

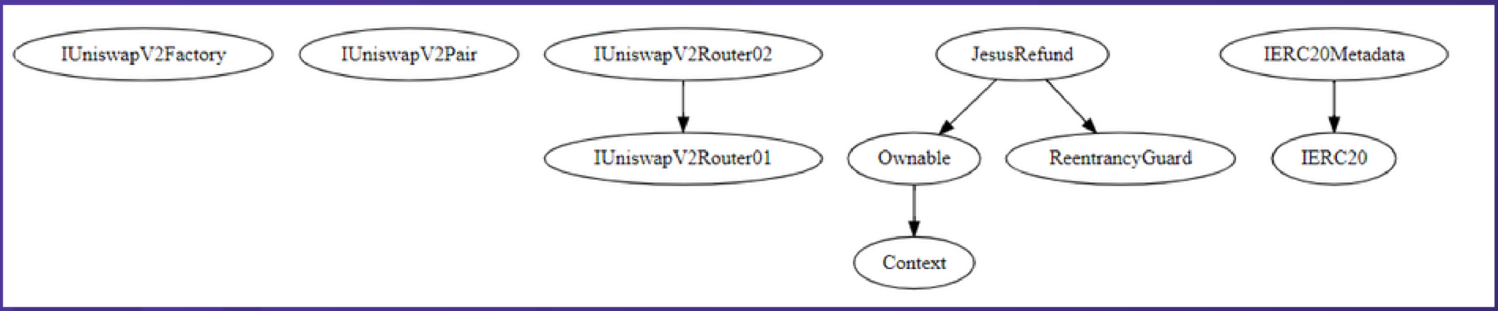
Low Risk

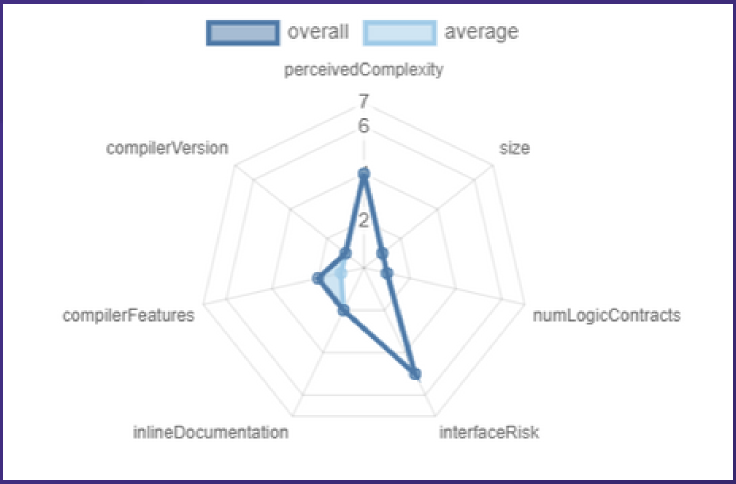
Issues on this level are minor details and warning that can remain unfixed.

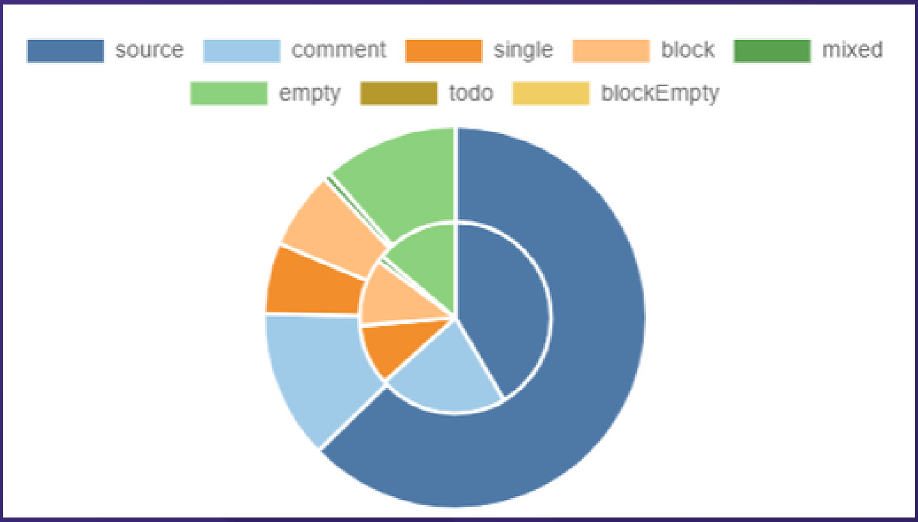
Informational

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

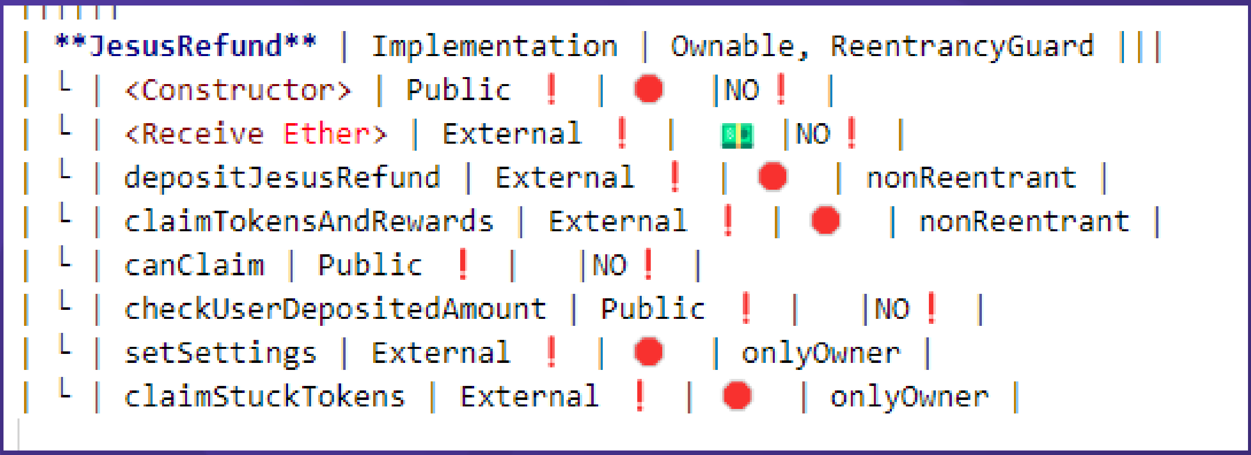
INHERITANCE TREES







FUNCTION DETAILS



```
| **JesusRefund** | Implementation | Ownable, ReentrancyGuard |||
| └ | <Constructor> | Public ❗️ | 🛑  |NO ❗️ |
| └ | <Receive Ether> | External ❗️ |  💵 |NO ❗️ |
| └ | depositJesusRefund | External ❗️ | 🛑  | nonReentrant |
| └ | claimTokensAndRewards | External ❗️ | 🛑  | nonReentrant |
| └ | canClaim | Public ❗️ |   |NO ❗️ |
| └ | checkUserDepositedAmount | Public ❗️ |   |NO ❗️ |
| └ | setSettings | External ❗️ | 🛑  | onlyOwner |
| └ | claimStuckTokens | External ❗️ | 🛑  | onlyOwner |
```

MANUAL REVIEW

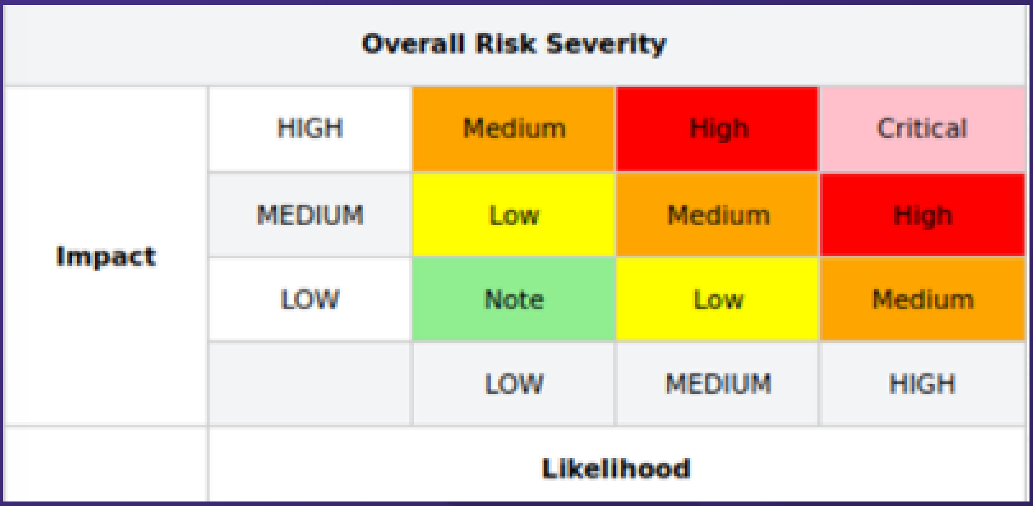
Severity Criteria

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	1
Medium Risk	🟠 Medium	0
Low Risk	🟡 Low	2
Suggestion & discussion	🔵 Informational	0
Gas Optimizations	🟣 Gas Opt.	0

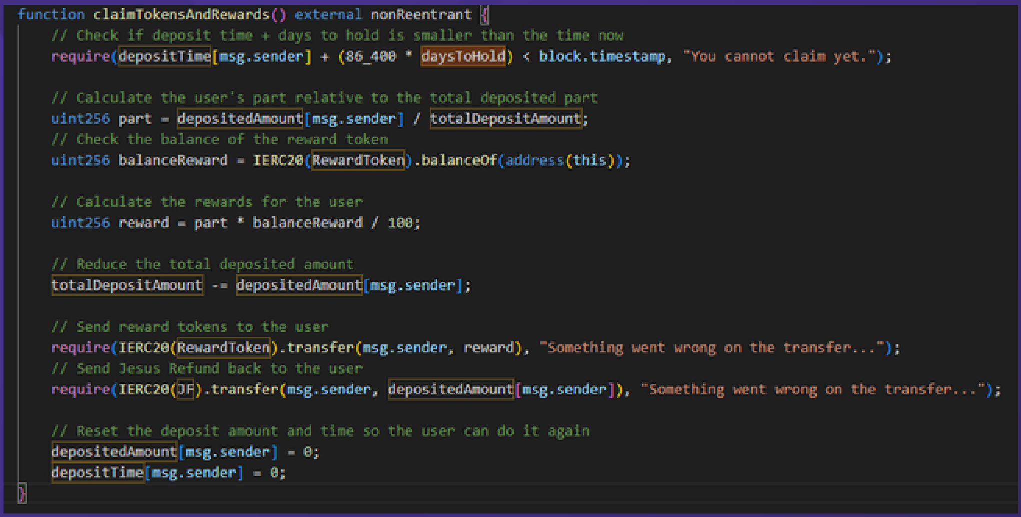
HIGH RISK FINDING

Reward Calculation error

Severity : HIGH

Overview

The **claimTokensAndRewards** function allows users to claim their tokens and rewards from a contract. It checks if the required holding period has passed, ensuring users cannot claim before the specified time It calculates the user's share based on their deposited amount relative to the total deposited amount. It checks the balance of the reward token held by the contract...



```
function claimTokensAndRewards() external nonReentrant {
    // Check if deposit time + days to hold is smaller than the time now
    require(depositTime[msg.sender] + (86_400 * daysToHold) < block.timestamp, "You cannot claim yet.");

    // Calculate the user's part relative to the total deposited part
    uint256 part = depositedAmount[msg.sender] / totalDepositAmount;
    // Check the balance of the reward token
    uint256 balanceReward = IERC20(RewardToken).balanceOf(address(this));

    // Calculate the rewards for the user
    uint256 reward = part * balanceReward / 100;

    // Reduce the total deposited amount
    totalDepositAmount -= depositedAmount[msg.sender];

    // Send reward tokens to the user
    require(IERC20(RewardToken).transfer(msg.sender, reward), "Something went wrong on the transfer...");
    // Send Jesus Refund back to the user
    require(IERC20(JF).transfer(msg.sender, depositedAmount[msg.sender]), "Something went wrong on the transfer...");

    // Reset the deposit amount and time so the user can do it again
    depositedAmount[msg.sender] = 0;
    depositTime[msg.sender] = 0;
}
```

Recommendation

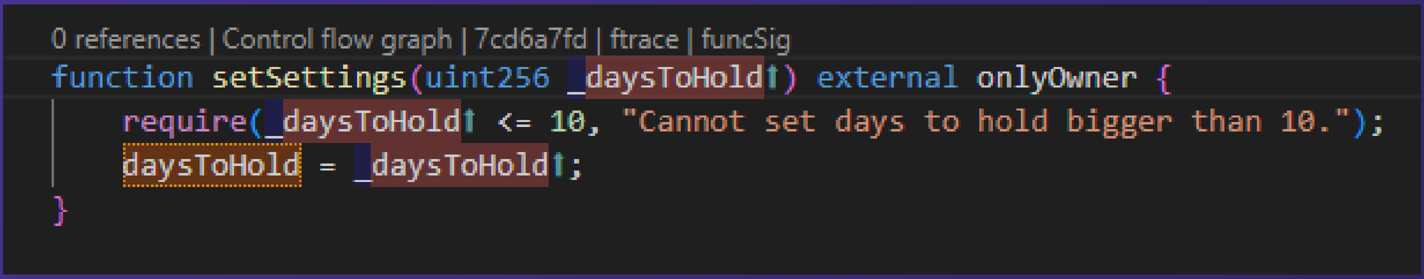
uint256 reward = part * balanceReward;, the rewards calculation is simply the product of the user's part (**part**) and the balance of the reward token (**balanceReward**). This calculation assumes that the balance of the reward token represents the total amount of rewards available. On the other hand, **uint256 reward = part * balanceReward / 100;** divides the reward calculation by **100**. This division by **100** assumes that the balance of the reward token represents the total amount of rewards available, but the reward distribution is intended to be in percentage form. By removing the division by **100**, the rewards calculation will not be scaled down to the appropriate proportion relative to the total rewards available. Therefore, if the intention is to distribute the rewards in proportion to the user's share without any scaling or percentage adjustment, **uint256 reward = part * balanceReward;** is the correct calculation to use.

LOW RISK FINDING

Owner can set hold date max 10

Severity : Low

Overview
The function takes a parameter **_daysToHold** of type **uint256**, representing the desired number of days to hold the tokens.
The line **require(_daysToHold <= 10, "Cannot set days to hold bigger than 10.");** checks if the **_daysToHold** value is less than or equal to 10. This check ensures that the owner cannot set an unreasonably high number of days.
Finally, if the requirement is met, the daysToHold variable i



```
0 references | Control flow graph | 7cd6a7fd | ftrace | funcSig
function setSettings(uint256 _daysToHold) external onlyOwner {
    require(_daysToHold <= 10, "Cannot set days to hold bigger than 10.");
    daysToHold = _daysToHold;
}
```

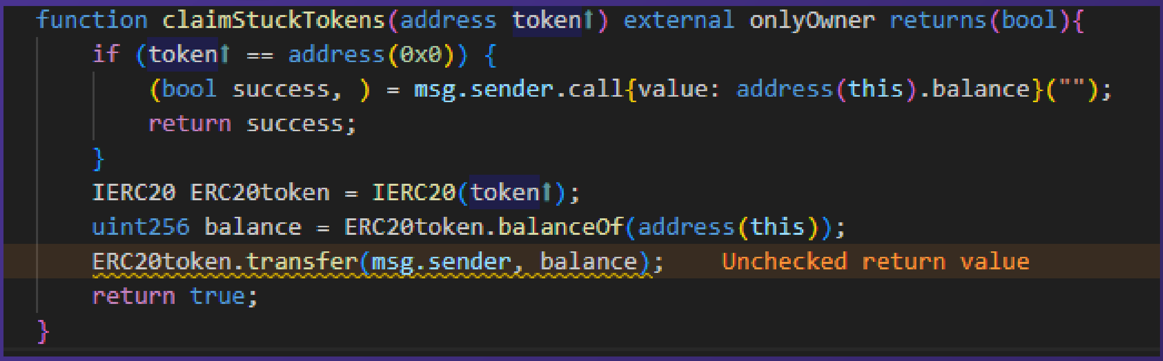
Recommendation
It is recommended to add additional access control measures, such as multi-factor authentication or time-based restrictions, to limit the number of authorized users who can call these functions. *Also check require message.*

LOW RISK FINDING

Owner can withdraw stuck tokens

Severity : Low

Overview
claimStuckTokens(), which allow the contract owner to withdraw locked tokens from the contract. The functions are properly restricted to only be executed by the contract owner.

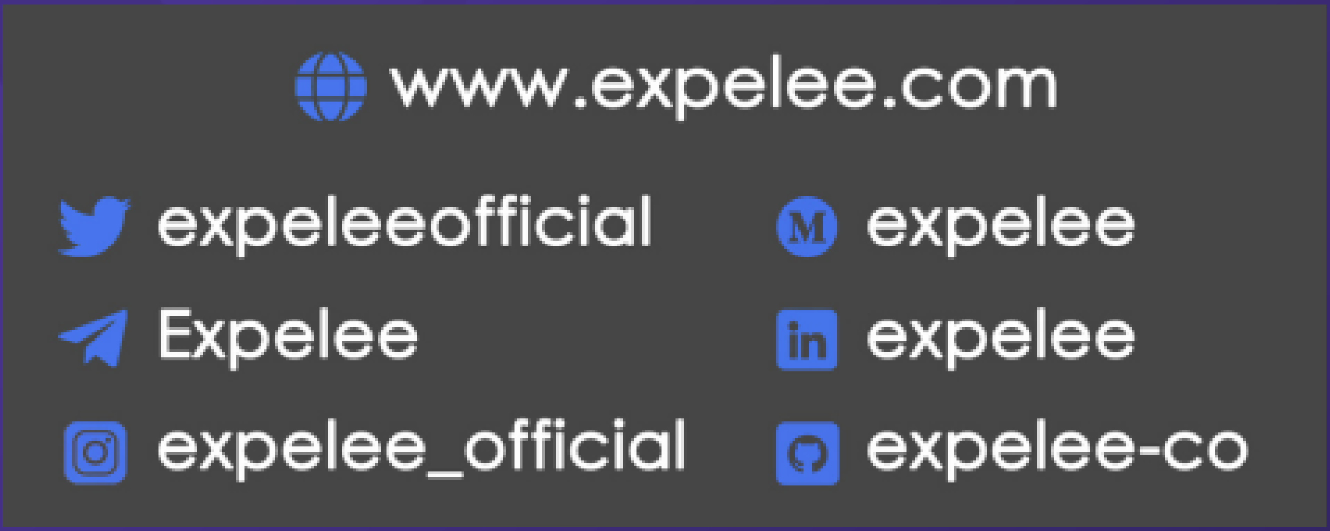


```
function claimStuckTokens(address token) external onlyOwner returns(bool){
    if (token == address(0x0)) {
        (bool success, ) = msg.sender.call{value: address(this).balance}("");
        return success;
    }
    IERC20 ERC20token = IERC20(token);
    uint256 balance = ERC20token.balanceOf(address(this));
    ERC20token.transfer(msg.sender, balance);    Unchecked return value
    return true;
}
```

Recommendation
While the functions are currently restricted to only be called by the contract owner, it is recommended to consider implementing a more robust access control mechanism.

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

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