expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Falcon404

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	4
Medium	1
Low	4
Informational	3

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner needs to enable trading ?	Yes, owner needs to enable trades
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	**Fail**
Audit Date	**9 March 2024**

CONTRACT DETAILS

Token Address: 0x509e2196c9786F49B574df838DcE18c0884f70a6

Name: Falcon404

Symbol: Falcon404

Decimals: 18

Network: BSC

Token Type:BEP-20

Owner: 0x5E67bdb740De026E91C49073f6967578D2894a0b

Deployer: 0x5E67bdb740De026E91C49073f6967578D2894a0b

Token Supply: 10000000000

Checksum: A2032c616934aeb47e6039f76b20d231

Testnet: --

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.

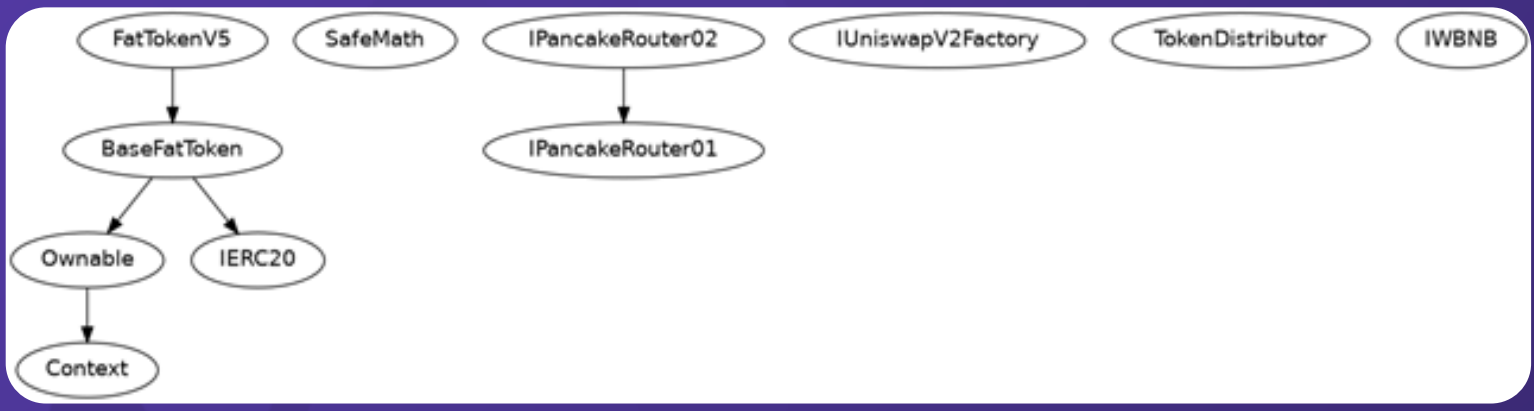
Low Risk

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



expelee

STATIC ANALYSIS

A static analysis of the code was performed using Slither. No issues were found.

```
INFO:Detectors:
FatTokenV5.swapTokenForFund(uint256,uint256) (FatTokenV5.sol#799-865) uses arbitrary from in transferFrom: _c.transferFrom(address(_tokenDistributor),addres
s(this),currencyBal) (FatTokenV5.sol#827-831)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#arbitrary-from-in-transferfrom
INFO:Detectors:
Reentrancy in FatTokenV5._transfer(address,address,uint256) (FatTokenV5.sol#615-725):
        External calls:
        - swapTokenForFund(numTokensSellToFund,swapFee) (FatTokenV5.sol#708)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,path,address(_tokenDistributor),block.timestamp
) (FatTokenV5.sol#811-821)
                - _c.transferFrom(address(_tokenDistributor),address(this),currencyBal) (FatTokenV5.sol#827-831)
                - IWBNB(currency).withdraw(toFundAmt) (FatTokenV5.sol#841)
                - _c.transfer(fundAddress,toFundAmt) (FatTokenV5.sol#844)
                - _swapRouter.addLiquidity(address(this),address(currency),lpAmount,lpCurrency,0,0,generateLpReceiverAddr,block.timestamp) (FatTokenV5.sol#8
58-863)
        External calls sending eth:
        - swapTokenForFund(numTokensSellToFund,swapFee) (FatTokenV5.sol#708)
                - fundAddress.transfer(toFundAmt) (FatTokenV5.sol#842)
        State variables written after the call(s):
        - _tokenTransfer(from,to,amount,takeFee,isSell,isTransfer) (FatTokenV5.sol#724)
                - _balances[sender] = _balances[sender] - tAmount (FatTokenV5.sol#748)
                - _balances[to] = _balances[to] + tAmount (FatTokenV5.sol#872)
        BaseFatToken._balances (FatTokenV5.sol#300) can be used in cross function reentrancies:
        - BaseFatToken._balances (FatTokenV5.sol#300)
        - FatTokenV5._basicTransfer(address,address,uint256) (FatTokenV5.sol#573-582)
        - FatTokenV5._takeTransfer(address,address,uint256) (FatTokenV5.sol#867-874)
        - FatTokenV5._tokenTransfer(address,address,uint256,bool,bool,bool) (FatTokenV5.sol#740-790)
        - FatTokenV5._transfer(address,address,uint256) (FatTokenV5.sol#615-725)
        - BaseFatToken.balanceOf(address) (FatTokenV5.sol#367-372)
        - FatTokenV5.constructor(string[],address[],uint256[],bool[]) (FatTokenV5.sol#436-517)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
FatTokenV5.swapTokenForFund(uint256,uint256) (FatTokenV5.sol#799-865) ignores return value by _c.transferFrom(address(_tokenDistributor),address(this),curre
ncyBal) (FatTokenV5.sol#827-831)
FatTokenV5.swapTokenForFund(uint256,uint256) (FatTokenV5.sol#799-865) ignores return value by _c.transfer(fundAddress,toFundAmt) (FatTokenV5.sol#844)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
```

```
INFO:Detectors:
FatTokenV5._transfer(address,address,uint256).isSell (FatTokenV5.sol#652) is a local variable never initialized
FatTokenV5._tokenTransfer(address,address,uint256,bool,bool,bool).feeAmount (FatTokenV5.sol#749) is a local variable never initialized
FatTokenV5._transfer(address,address,uint256).takeFee (FatTokenV5.sol#651) is a local variable never initialized
FatTokenV5._transfer(address,address,uint256).isTransfer (FatTokenV5.sol#719) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
TokenDistributor.constructor(address) (FatTokenV5.sol#416-418) ignores return value by IERC20(token).approve(msg.sender,uint256(~ uint256(0))) (FatTokenV5.s
ol#417)
FatTokenV5.constructor(string[],address[],uint256[],bool[]) (FatTokenV5.sol#436-517) ignores return value by IERC20(currency).approve(address(swapRouter),MA
X) (FatTokenV5.sol#489)
FatTokenV5.swapTokenForFund(uint256,uint256) (FatTokenV5.sol#799-865) ignores return value by _swapRouter.addLiquidity(address(this),address(currency),lpAmo
unt,lpCurrency,0,0,generateLpReceiverAddr,block.timestamp) (FatTokenV5.sol#850-863)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
BaseFatToken.allowance(address,address).owner (FatTokenV5.sol#375) shadows:
        - Ownable.owner() (FatTokenV5.sol#71-73) (function)
BaseFatToken._approve(address,address,uint256).owner (FatTokenV5.sol#389) shadows:
        - Ownable.owner() (FatTokenV5.sol#71-73) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
BaseFatToken.changeSwapLimit(uint256) (FatTokenV5.sol#310-312) should emit an event for:
        - maxBuyAmount = _maxBuyAmount (FatTokenV5.sol#311)
BaseFatToken.changeWalletLimit(uint256) (FatTokenV5.sol#314-316) should emit an event for:
        - maxWalletAmount = _amount (FatTokenV5.sol#315)
BaseFatToken.completeCustoms(uint256[]) (FatTokenV5.sol#335-350) should emit an event for:
        - _buyLPFee = customs[0] (FatTokenV5.sol#337)
        - _buyBurnFee = customs[1] (FatTokenV5.sol#338)
        - _buyFundFee = customs[2] (FatTokenV5.sol#339)
        - _sellLPFee = customs[3] (FatTokenV5.sol#341)
        - _sellBurnFee = customs[4] (FatTokenV5.sol#342)
        - _sellFundFee = customs[5] (FatTokenV5.sol#343)
FatTokenV5.setkb(uint256) (FatTokenV5.sol#555-557) should emit an event for:
        - kb = a (FatTokenV5.sol#556)
FatTokenV5.setAirdropNumbs(uint256) (FatTokenV5.sol#586-589) should emit an event for:
        - airdropNumbs = newValue (FatTokenV5.sol#588)
FatTokenV5.setNumTokensSellRate(uint256) (FatTokenV5.sol#604-607) should emit an event for:
        - numTokensSellRate = newValue (FatTokenV5.sol#606)
FatTokenV5.setSwapAtAmount(uint256) (FatTokenV5.sol#611-613) should emit an event for:
```

expelee

STATIC ANALYSIS

```
INFO:Detectors:
Reentrancy in FatTokenV5._transfer(address,address,uint256) (FatTokenV5.sol#615-725):
        External calls:
        - swapTokenForFund(numTokensSellToFund,swapFee) (FatTokenV5.sol#708)
                - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,path,address(_tokenDistributor),block.timestamp
) (FatTokenV5.sol#811-821)
                - _c.transferFrom(address(_tokenDistributor),address(this),currencyBal) (FatTokenV5.sol#827-831)
                - IWBNB(currency).withdraw(toFundAmt) (FatTokenV5.sol#841)
                - _c.transfer(fundAddress,toFundAmt) (FatTokenV5.sol#844)
                - _swapRouter.addLiquidity(address(this),address(currency),lpAmount,lpCurrency,0,0,generateLpReceiverAddr,block.timestamp) (FatTokenV5.sol#8
50-863)
        External calls sending eth:
        - swapTokenForFund(numTokensSellToFund,swapFee) (FatTokenV5.sol#708)
                - fundAddress.transfer(toFundAmt) (FatTokenV5.sol#842)
        Event emitted after the call(s):
        - Transfer(sender,to,tAmount) (FatTokenV5.sol#873)
                - _tokenTransfer(from,to,amount,takeFee,isSell,isTransfer) (FatTokenV5.sol#724)
Reentrancy in FatTokenV5.swapTokenForFund(uint256,uint256) (FatTokenV5.sol#799-865):
        External calls:
        - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,path,address(_tokenDistributor),block.timestamp) (FatTo
kenV5.sol#811-821)
        Event emitted after the call(s):
        - Failed_swapExactTokensForTokensSupportingFeeOnTransferTokens() (FatTokenV5.sol#820)
Reentrancy in FatTokenV5.swapTokenForFund(uint256,uint256) (FatTokenV5.sol#799-865):
        External calls:
        - _swapRouter.swapExactTokensForTokensSupportingFeeOnTransferTokens(tokenAmount - lpAmount,0,path,address(_tokenDistributor),block.timestamp) (FatTo
kenV5.sol#811-821)
        - _c.transferFrom(address(_tokenDistributor),address(this),currencyBal) (FatTokenV5.sol#827-831)
        - IWBNB(currency).withdraw(toFundAmt) (FatTokenV5.sol#841)
        - _c.transfer(fundAddress,toFundAmt) (FatTokenV5.sol#844)
        - _swapRouter.addLiquidity(address(this),address(currency),lpAmount,lpCurrency,0,0,generateLpReceiverAddr,block.timestamp) (FatTokenV5.sol#850-863)
        External calls sending eth:
        - fundAddress.transfer(toFundAmt) (FatTokenV5.sol#842)
        Event emitted after the call(s):
        - Failed_AddLiquidity() (FatTokenV5.sol#862)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
Variable IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (FatTokenV5.sol#193) is too similar t
o IPancakeRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (FatTokenV5.sol#194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
BaseFatToken.deadAddress (FatTokenV5.sol#297) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
BaseFatToken._mainPair (FatTokenV5.sol#308) should be immutable
BaseFatToken._swapRouter (FatTokenV5.sol#304) should be immutable
BaseFatToken.currency (FatTokenV5.sol#275) should be immutable
BaseFatToken.currencyIsEth (FatTokenV5.sol#264) should be immutable
BaseFatToken.decimals (FatTokenV5.sol#294) should be immutable
BaseFatToken.enableKillBlock (FatTokenV5.sol#267) should be immutable
BaseFatToken.enableOffTrade (FatTokenV5.sol#266) should be immutable
BaseFatToken.enableRewardList (FatTokenV5.sol#268) should be immutable
BaseFatToken.totalSupply (FatTokenV5.sol#295) should be immutable
FatTokenV5._tokenDistributor (FatTokenV5.sol#428) should be immutable
FatTokenV5.enableTransferFee (FatTokenV5.sol#591) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:FatTokenV5.sol analyzed (11 contracts with 93 detectors), 80 result(s) found
```

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			

HIGH RISK FINDING

Centralization – Enabling Trades

Severity: High

function: Launch

Status: Open

Overview:

The Launch function permits only the contract owner to activate trading capabilities. Until this function is executed, no investors can buy, sell, or transfer their tokens. This places a high degree of control and centralization in the hands of the contract owner.

```
function launch() external onlyOwner {
require(startTradeBlock == 0, "already started");
  startTradeBlock = block.number;
}
```

Suggestion

To reduce centralization and potential manipulation, consider one of the following approaches:

1.Automatically enable trading after a specified condition, such as the completion of a presale, is met.

2.If manual activation is still desired, consider transferring the ownership of the contract to a trustworthy, third-party entity like a certified "PinkSale Safu" developer. This can give investors more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad-faith actions by the original owner.

HIGH RISK FINDING

Centralization – The owner can Blacklist Wallet

Severity: High

function: multi bclist

Status: Open

Overview:

The owner can blacklist multiple wallets.

```
 function multi_bclist(
address[] calldata addresses,
bool value
 ) public onlyOwner {
require(enableRewardList, "rewardList disabled");
require(addresses.length < 201);
for (uint256 i; i < addresses.length; ++i) {
    _rewardList[addresses[i]] = value;
  }
 }
```

HIGH RISK FINDING

Centralization – The owner can lock the token

Severity: High

function: setMaxWalletAmount

Status: Open

Overview:

In this changeWalletLimit.

```
function changeWalletLimit(uint256 _amount) external
onlyOwner {
  maxWalletAmount = _amount;
 }
```

Suggestion

It is recommended that there be a required check for zero address.

HIGH RISK FINDING

Centralization – Unsafe Usage of tx.origin

Severity: High

function: Tx.origin

Status: Open

Overview:

Avoid using TX.origin for authorization, another contract can have a method that will call your contract (where the user has some funds for instance) and your contract will authorize that transaction as your address is in tx. origin.

```
_feeWhiteList[tx.origin] = true;
}
```

Suggestion

You should use msg. sender for authorization (if another contract calls your contract msg.sender will be the address of the contract and not the address of the user who called the contract.

LOW RISK FINDING

Centralization –Liquidity is added to EOA

Severity: Medium

function: addLiquidity

Status: Open

Overview:

Liquidity is added to EOA. It may be drained by the generateLpReceiverAddr.

```
 if (lpAmount > 0 && lpCurrency > 0) {
try
    _swapRouter.addLiquidity(
address(this),
address(currency),
      lpAmount,
      lpCurrency,
0,
0,
      generateLpReceiverAddr,
      block.timestamp
    )
```

Suggestion

It is suggested that the address should be a contract address or a dead address.

LOW RISK FINDING

Centralization – Missing Zero Address

Severity: Low

function: excludeFromLimits

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.

```
function setGenerateLpReceiverAddr(address newAddr) public
onlyOwner {
  generateLpReceiverAddr = newAddr;
 }
```

Suggestion

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

LOW RISK FINDING

Centralization – Missing Events

Severity: Low

function: Missing Events

Status: Open

Overview:

They serve as a mechanism for emitting and recording data onto the blockchain, making it transparent and easily accessible.

```
  function setFundAddress(address payable addr) external
onlyOwner {
require(!isContract(addr), "fundaddress is a contract ");
   fundAddress = addr;
   _feeWhiteList[addr] = true;
 }
function setAirdropNumbs(uint256 newValue) public onlyOwner {
require(newValue <= 3, "newValue must <= 3");
   airdropNumbs = newValue;
 }
function setNumTokensSellRate(uint256 newValue) public onlyOwner
{
require(newValue != 0, "greater than 0");
   numTokensSellRate = newValue;
 }

function setSwapAtAmount(uint256 newValue) public onlyOwner {
   swapAtAmount = newValue;
 }
function setTransferFee(uint256 newValue) public onlyOwner {
require(newValue <= 2500, "transfer > 25 !");
   transferFee = newValue;
 }
```

LOW RISK FINDING

```
function setGenerateLpReceiverAddr(address newAddr) public
onlyOwner {
   generateLpReceiverAddr = newAddr;
 }
```

Suggestion

Add an event to these important functions where address updation is happening. This can also be marked as an indexed event for better off-chain tracking.

LOW RISK FINDING

Centralization – Local Variable Shadowing

Severity: Low

function: Shadowing Local

Status: Open

Overview:

```
function allowance(address owner, address spender) public view
override returns (uint256) {
return _allowances[owner][spender];
}
function _approve(address owner, address spender, uint256
amount) private {
require(owner != address(0), "BEP20: approve from the zero
address");
require(spender != address(0), "BEP20: approve to the zero address");
_allowances[owner][spender] = amount;
emit Approval(owner, spender, amount);
}
```

Suggestion

Rename the local variable that shadows another component.

LOW RISK FINDING

Centralization – Missing Threshhold

Severity: Low

function: Missing Threshhold

Status: Open

Overview:

```
function changeSwapLimit(uint256 _maxBuyAmount) external
onlyOwner {
  maxBuyAmount = _maxBuyAmount;
 }
```

Suggestion

It is recommended that there should be a threshhold limit for changeswaplimit.

INFORMATIONAL & OPTIMIZATIONS

Optimization
Severity: Informational
subject: Remove Safe Math
Status: Open
Line: 84-146

Overview:
compiler version above 0.8.0 can control arithmetic overflow/underflow, it is recommended to remove the unwanted code to avoid high gas fees.

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: Informational

subject: Floating Pragma

Status: Open

Overview:

It is considered best practice to pick one compiler version and stick with it. With a floating pragma, contracts may accidentally be deployed using an outdated.

```
pragma solidity ^0.8.4;
```

Suggestion:

Adding the latest constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: Optimization

subject: Remove Unused Code

Status: Open

Overview:

Unused variables are allowed in Solidity, and they do. not pose a direct security issue. It is the best practice though to avoid them.

```
function _msgData() internal view returns (bytes memory) {
  this; // silence state mutability warning without generating
bytecode - see
https://github.com/ethereum/solidity/issues/2691
return msg.data;
 }
```

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.