expelee

Building the Futuristic Blockchain Ecosystem

# SECURITY AUDIT REPORT

ZMLM

ZMLM.FINANCE

# 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 | Not Done |
| Audit Date | 17 April 2023 |

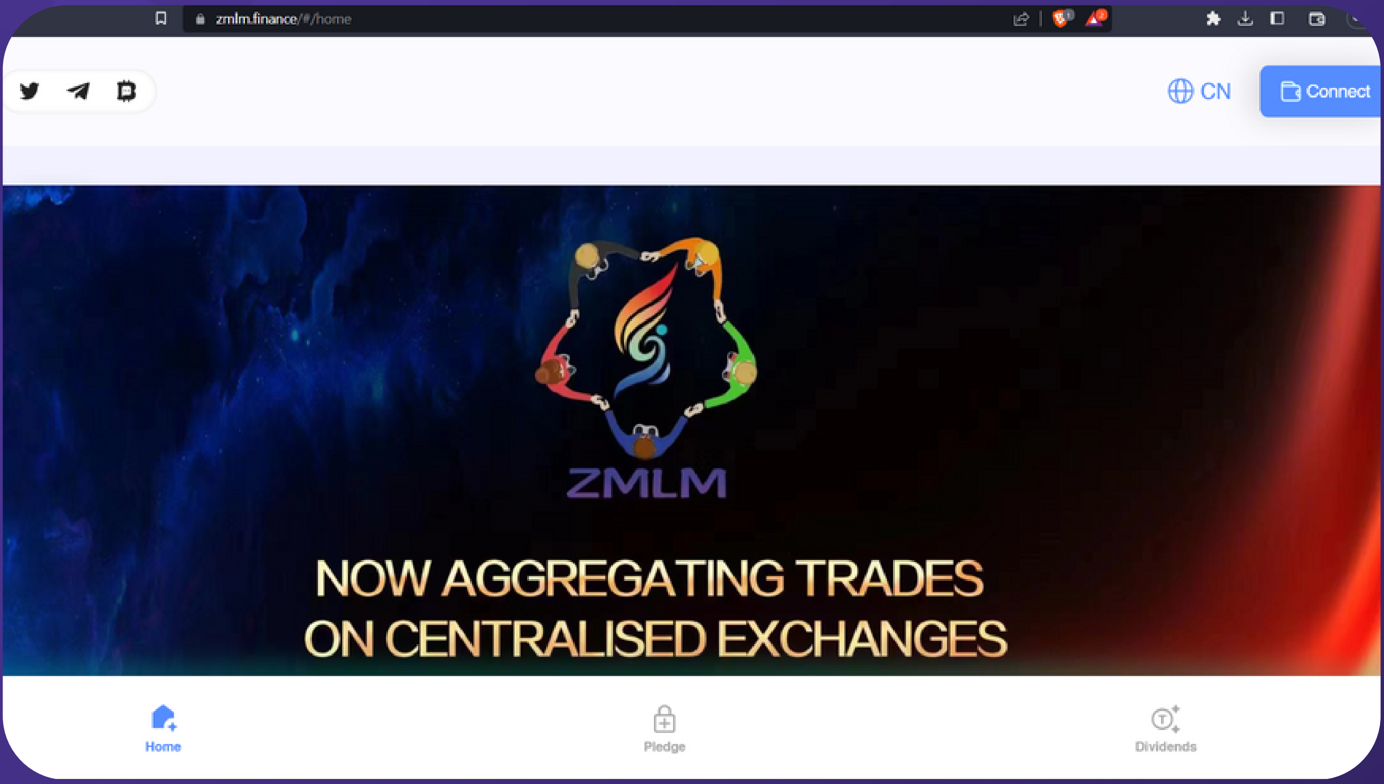
# PROJECT DESCRIPTION

## ZMLM.FINANCE

NOW AGGREGATING TRADES ON CENTRALISED EXCHANGES
THE FUTURE OF DECENTRALISED EXCHANGE
ON-CHAIN AGGREGATION OF TRANSACTIONS

# SOCIAL MEDIA PROFILES

## ZMLM.FINANCE



https://t.me/zm_lmsq

https://twitter.com/zm_Government

https://zmlm.finance/#/home

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

Team Expelee

# CONTRACT DETAILS

Token Name: Zmlm.FInance

Symbol: ZM

Network: Binance Smart Chain

Language: Solidity

Contract Address:
0x44ac762dB7E7170A48e895fDC81Bc2e81c188888

Total Supply: 7999

Contract SHA-256 Checksum: -

Owner's Wallet:
0x000000000000000000000000000000000000dEaD

Deployer's Wallet:
0x0385B79200e5aaC080A170E1cf89672749fc6999

Testnet:
https://testnet.bscscan.com/address/0xe427c4d8ed5c473 3d9ff9c56b531652671ea6566

## OWNER PRIVILEGES

- Owner can set NFT reward condition without limit
- Owner can set large NFT lp condition without limit
- Owner can set large NFT team lp condition without limit
- Owner can set little NFT lp condition without limit
- Owner can set little NFT team lp condition without limit
- Auto liquidity is going to an externally owned account
- Owner can add new swap pair
- Owner can exclude accounts from fees
- Owner can set whitelist account for NFT
- Current buy fees 1.3% and  sell fees 2.8% also a fee rate for removing liquidity 0.3% it can not be change
- Owner can withdraw any token from the contract
- Owner can set rewardGas

# 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.

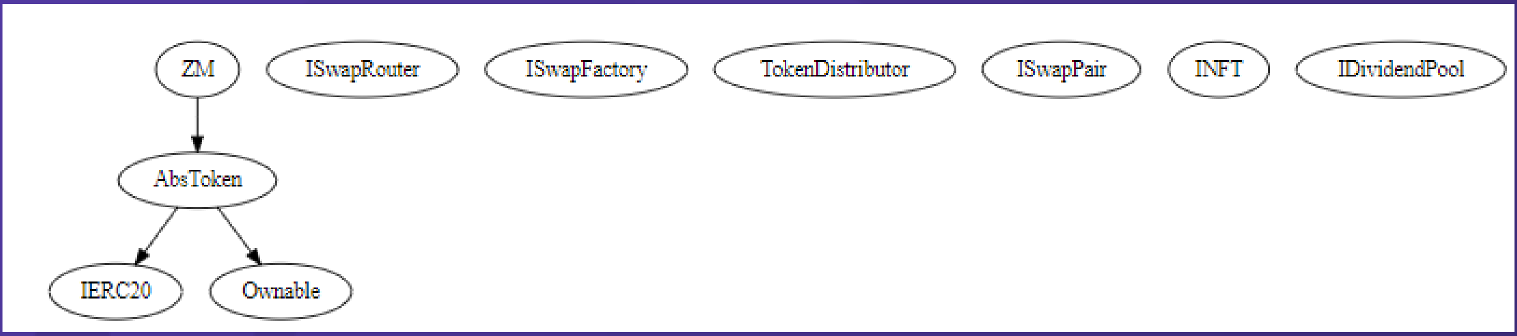
## Low Risk

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

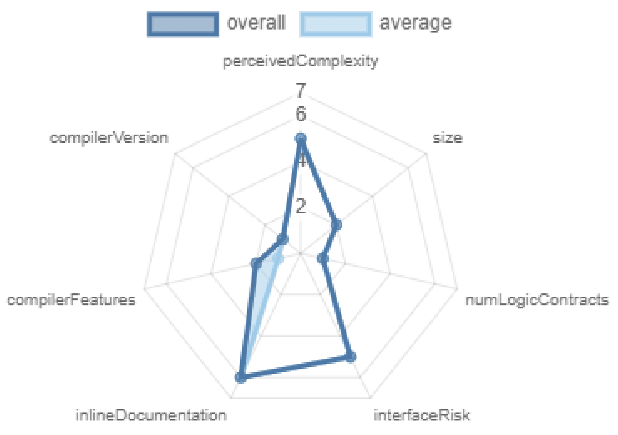
## Informational

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

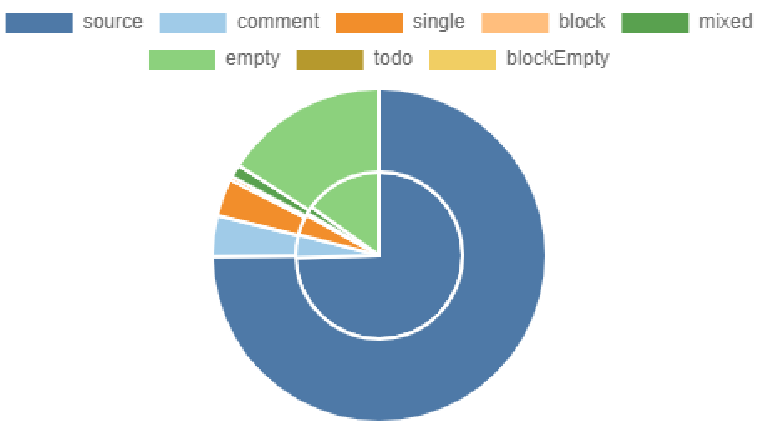
# INHERITANCE TREES



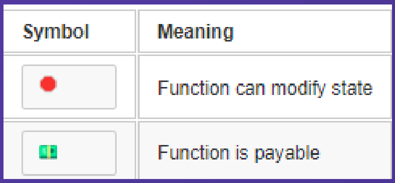
## Risk



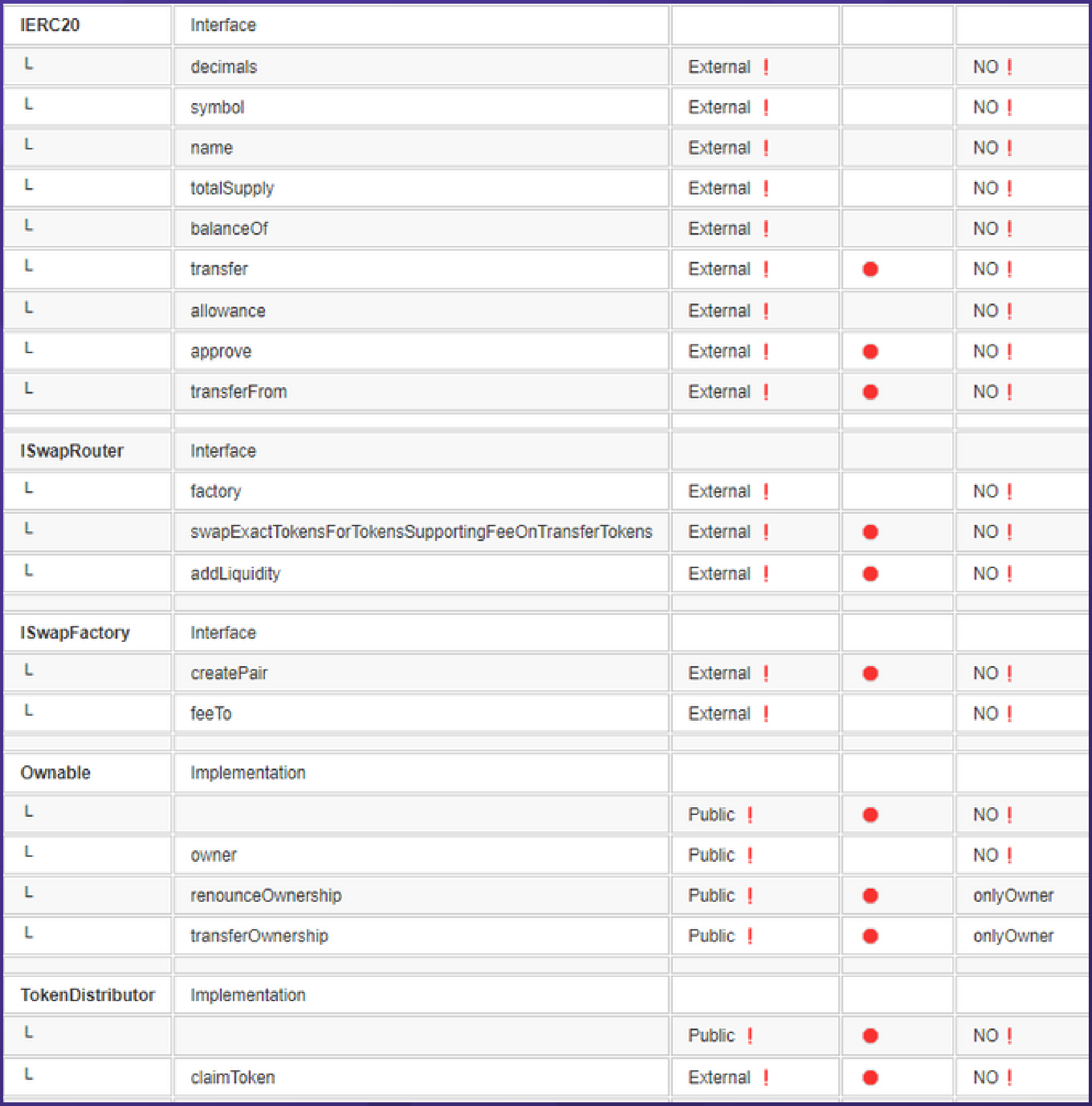
## Source Lines (sloc vs. nsloc)



# FUNCTION DETAILS



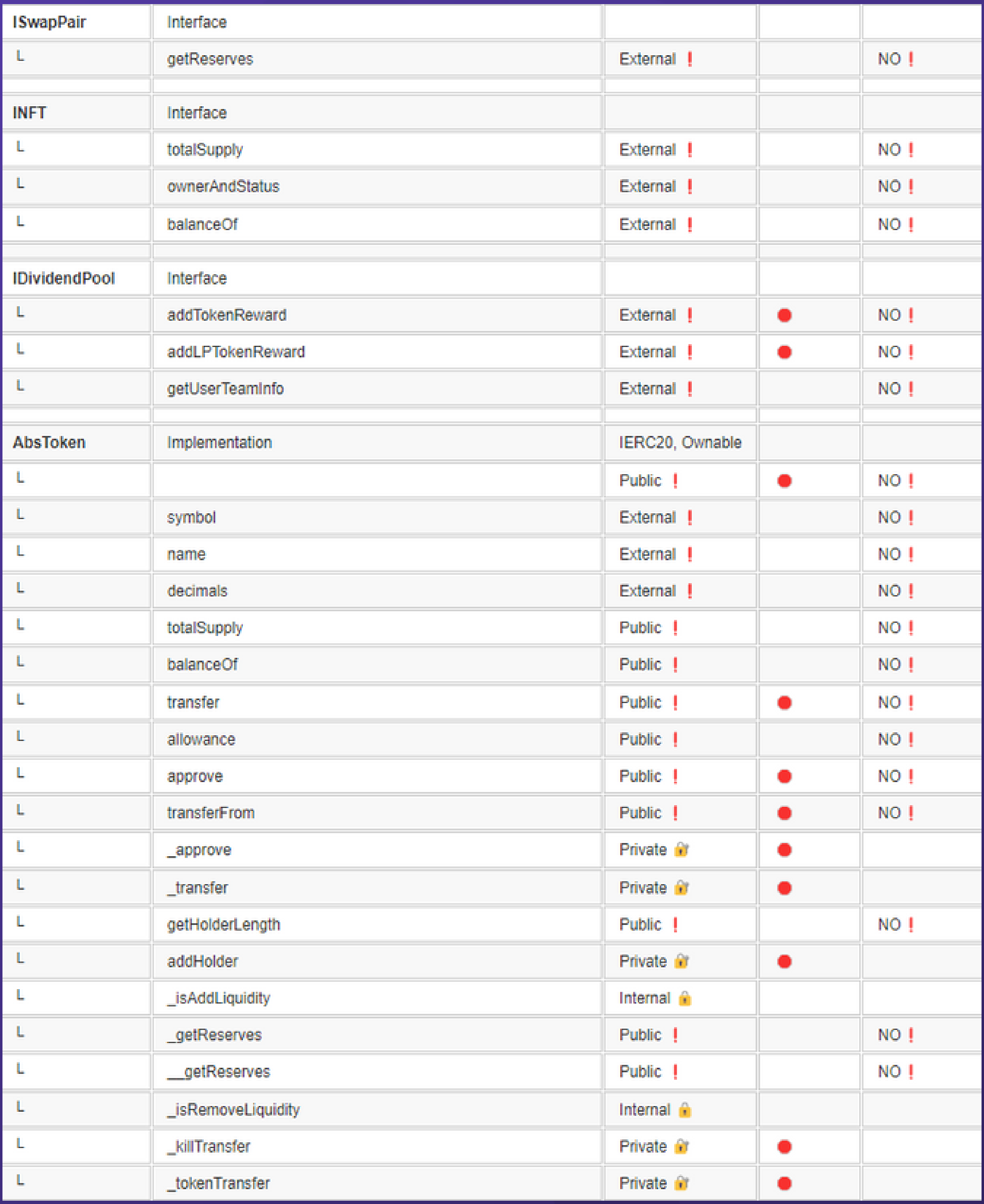
| Symbol | Meaning |
|---|---|
| ● | Function can modify state |
| 💵 | Function is payable |



| | | | | |
|---|---|---|---|---|
| IERC20 | Interface | | | |
| └ | decimals | External ❗ | | NO ❗ |
| └ | symbol | External ❗ | | NO ❗ |
| └ | name | External ❗ | | NO ❗ |
| └ | totalSupply | External ❗ | | NO ❗ |
| └ | balanceOf | External ❗ | | NO ❗ |
| └ | transfer | External ❗ | ● | NO ❗ |
| └ | allowance | External ❗ | | NO ❗ |
| └ | approve | External ❗ | ● | NO ❗ |
| └ | transferFrom | External ❗ | ● | NO ❗ |
| | | | | |
| ISwapRouter | Interface | | | |
| └ | factory | External ❗ | | NO ❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | ● | NO ❗ |
| └ | addLiquidity | External ❗ | ● | NO ❗ |
| | | | | |
| ISwapFactory | Interface | | | |
| └ | createPair | External ❗ | ● | NO ❗ |
| └ | feeTo | External ❗ | | NO ❗ |
| | | | | |
| Ownable | Implementation | | | |
| └ | | Public ❗ | ● | NO ❗ |
| └ | owner | Public ❗ | | NO ❗ |
| └ | renounceOwnership | Public ❗ | ● | onlyOwner |
| └ | transferOwnership | Public ❗ | ● | onlyOwner |
| | | | | |
| TokenDistributor | Implementation | | | |
| └ | | Public ❗ | ● | NO ❗ |
| └ | claimToken | External ❗ | ● | NO ❗ |

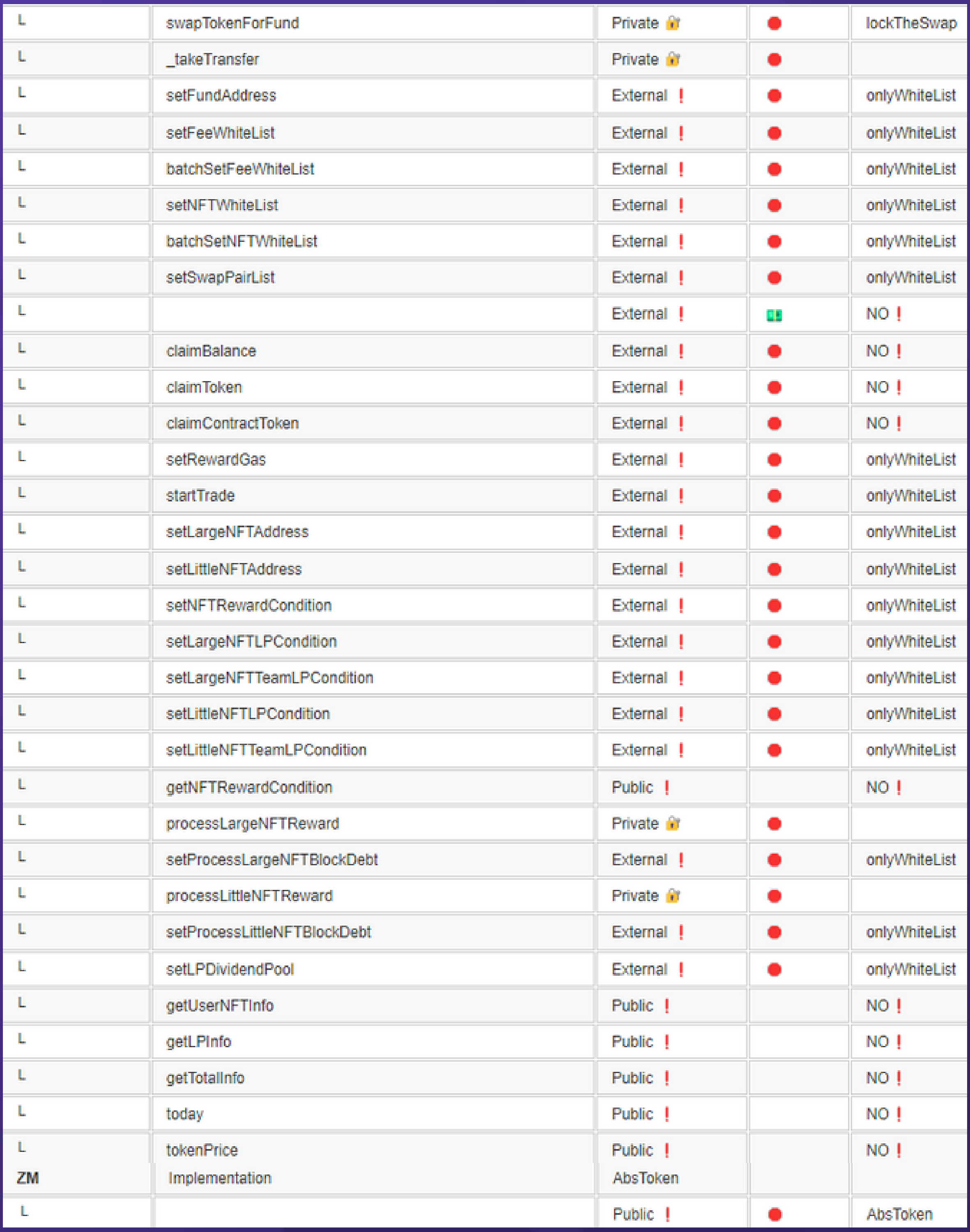
expelee

# FUNCTION DETAILS



| | | | | |
|---|---|---|---|---|
| **ISwapPair** | Interface | | | |
| └ | getReserves | External ❗ | | NO ❗ |
| | | | | |
| **INFT** | Interface | | | |
| └ | totalSupply | External ❗ | | NO ❗ |
| └ | ownerAndStatus | External ❗ | | NO ❗ |
| └ | balanceOf | External ❗ | | NO ❗ |
| | | | | |
| **IDividendPool** | Interface | | | |
| └ | addTokenReward | External ❗ | ● | NO ❗ |
| └ | addLPTokenReward | External ❗ | ● | NO ❗ |
| └ | getUserTeamInfo | External ❗ | | NO ❗ |
| | | | | |
| **AbsToken** | Implementation | IERC20, Ownable | | |
| └ | | Public ❗ | ● | NO ❗ |
| └ | symbol | External ❗ | | NO ❗ |
| └ | name | External ❗ | | NO ❗ |
| └ | decimals | External ❗ | | NO ❗ |
| └ | totalSupply | Public ❗ | | NO ❗ |
| └ | balanceOf | Public ❗ | | NO ❗ |
| └ | transfer | Public ❗ | ● | NO ❗ |
| └ | allowance | Public ❗ | | NO ❗ |
| └ | approve | Public ❗ | ● | NO ❗ |
| └ | transferFrom | Public ❗ | ● | NO ❗ |
| └ | _approve | Private 🔐 | ● | |
| └ | _transfer | Private 🔐 | ● | |
| └ | getHolderLength | Public ❗ | | NO ❗ |
| └ | addHolder | Private 🔐 | ● | |
| └ | _isAddLiquidity | Internal 🔒 | | |
| └ | _getReserves | Public ❗ | | NO ❗ |
| └ | __getReserves | Public ❗ | | NO ❗ |
| └ | _isRemoveLiquidity | Internal 🔒 | | |
| └ | _killTransfer | Private 🔐 | ● | |
| └ | _tokenTransfer | Private 🔐 | ● | |

# FUNCTION DETAILS



| | | | | |
|---|---|---|---|---|
| L | swapTokenForFund | Private 🔐 | ● | lockTheSwap |
| L | _takeTransfer | Private 🔐 | ● | |
| L | setFundAddress | External ❗ | ● | onlyWhiteList |
| L | setFeeWhiteList | External ❗ | ● | onlyWhiteList |
| L | batchSetFeeWhiteList | External ❗ | ● | onlyWhiteList |
| L | setNFTWhiteList | External ❗ | ● | onlyWhiteList |
| L | batchSetNFTWhiteList | External ❗ | ● | onlyWhiteList |
| L | setSwapPairList | External ❗ | ● | onlyWhiteList |
| L | | External ❗ | 💵 | NO ❗ |
| L | claimBalance | External ❗ | ● | NO ❗ |
| L | claimToken | External ❗ | ● | NO ❗ |
| L | claimContractToken | External ❗ | ● | NO ❗ |
| L | setRewardGas | External ❗ | ● | onlyWhiteList |
| L | startTrade | External ❗ | ● | onlyWhiteList |
| L | setLargeNFTAddress | External ❗ | ● | onlyWhiteList |
| L | setLittleNFTAddress | External ❗ | ● | onlyWhiteList |
| L | setNFTRewardCondition | External ❗ | ● | onlyWhiteList |
| L | setLargeNFTLPCondition | External ❗ | ● | onlyWhiteList |
| L | setLargeNFTTeamLPCondition | External ❗ | ● | onlyWhiteList |
| L | setLittleNFTLPCondition | External ❗ | ● | onlyWhiteList |
| L | setLittleNFTTeamLPCondition | External ❗ | ● | onlyWhiteList |
| L | getNFTRewardCondition | Public ❗ | | NO ❗ |
| L | processLargeNFTReward | Private 🔐 | ● | |
| L | setProcessLargeNFTBlockDebt | External ❗ | ● | onlyWhiteList |
| L | processLittleNFTReward | Private 🔐 | ● | |
| L | setProcessLittleNFTBlockDebt | External ❗ | ● | onlyWhiteList |
| L | setLPDividendPool | External ❗ | ● | onlyWhiteList |
| L | getUserNFTInfo | Public ❗ | | NO ❗ |
| L | getLPInfo | Public ❗ | | NO ❗ |
| L | getTotalInfo | Public ❗ | | NO ❗ |
| L | today | Public ❗ | | NO ❗ |
| L | tokenPrice | Public ❗ | | NO ❗ |
| ZM | Implementation | AbsToken | | |
| L | | Public ❗ | ● | AbsToken |

# MANUAL REVIEW

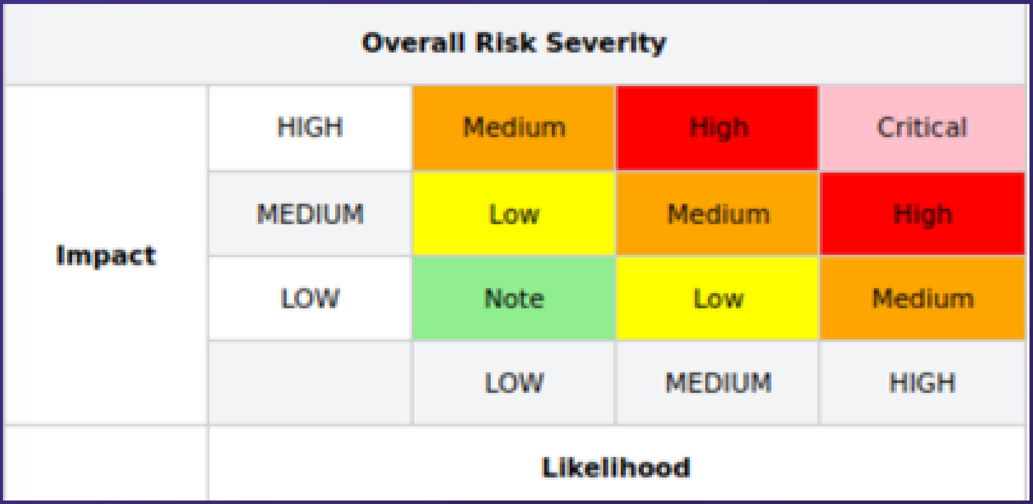
## Severity Criteria

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 | 5 |
| Medium Risk | ● Medium | 2 |
| Low Risk | ● Low | 3 |
| Suggestion & discussion | ● Informational | 1 |
| Gas Optimizations | ● Gas Opt. | 0 |

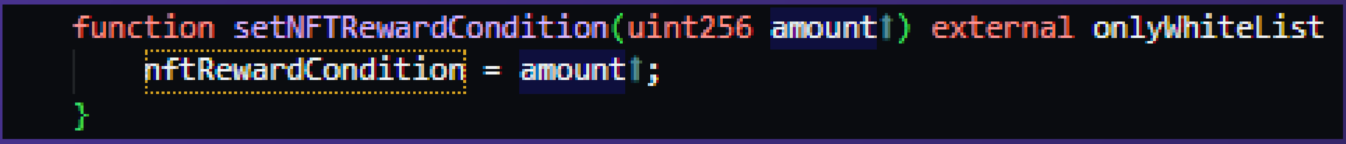
# HIGH RISK FINDING

## Owner can set NFT reward condition without limit

**Severity : High**

### Overview

There is no mechanism in place to prevent the contract owner from setting the **nftRewardCondition** to **zero**, which would effectively disable the reward distribution altogether. This could be problematic if users have purchased the NFTs with the expectation of receiving rewards. If the **nftRewardCondition** is set too high, it might become difficult or impossible for the contract to meet the condition, and this could lead to a situation where NFT holders are unable to claim their rewards



```
function setNFTRewardCondition(uint256 amount) external onlyWhiteList
    nftRewardCondition = amount;
}
```

### Recommendation

I would recommend adding a check in the **setNFTRewardCondition** function to prevent the **nftRewardCondition** from being set to **zero.** it's important to set a reasonable **nftRewardCondition** value that is achievable by the contract and that does not create an excessive accumulation of funds

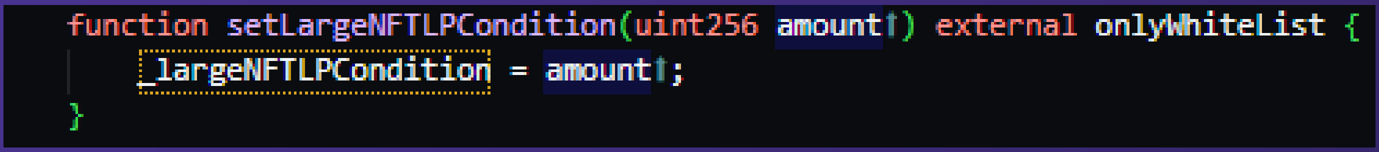
# HIGH RISK FINDING

## Owner can set large NFT lp condition without limit

**Severity : High**

### Overview

**_largeNFTLPCondition** value, which is used as a condition to determine whether a user is eligible to receive rewards for holding a specific NFT. If the owner sets a very high value for **_largeNFTLPCondition**, it could potentially make it very difficult for users to meet the condition and receive rewards.



```
function setLargeNFTLPCondition(uint256 amount) external onlyWhiteList {
    _largeNFTLPCondition = amount;
}
```

### Recommendation

I would recommend that the owner set a reasonable value for **_largeNFTLPCondition** that is achievable by users who hold the NFT. The value should be based on the current state of the market and should not be so high as to make it virtually impossible for users to qualify for rewards.

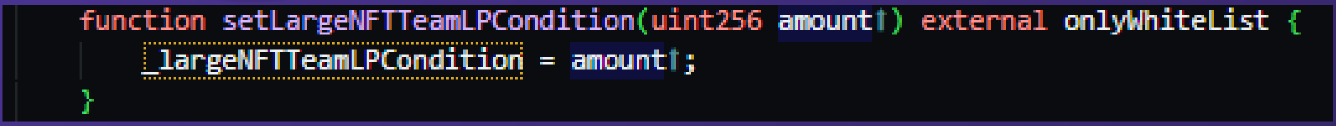
# HIGH RISK FINDING

**Owner can set large NFT team lp condition without limit**

**Severity : High**

**Overview**

**_largeNFTTeamLPCondition** value, which is used as a condition to determine whether a user is eligible to receive rewards for holding a specific NFT. If the owner sets a very high value for **_largeNFTTeamLPCondition**, it could potentially make it very difficult for users to meet the condition and receive rewards.



```
function setLargeNFTTeamLPCondition(uint256 amount) external onlyWhiteList {
    _largeNFTTeamLPCondition = amount;
}
```

**Recommendation**

I would recommend that the owner set a reasonable value for **_largeNFTTeamLPCondition** that is achievable by users who hold the NFT. The value should be based on the current state of the market and should not be so high as to make it virtually impossible for users to qualify for rewards.

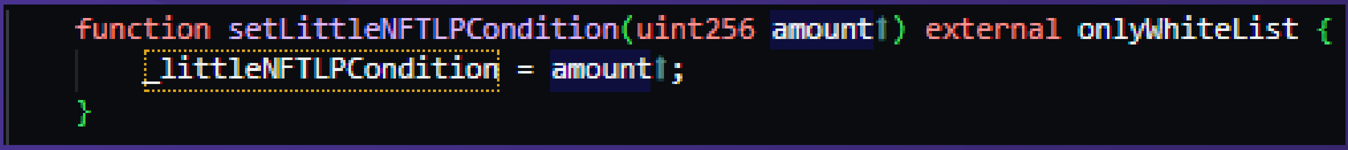
# HIGH RISK FINDING

## Owner can set little NFT lp condition without limit

**Severity : High**

**Overview**

**_littleNFTLPCondition** value, which is used as a condition to determine whether a user is eligible to receive rewards for holding a specific NFT. If the owner sets a very high value for **_littleNFTLPCondition** , it could potentially make it very difficult for users to meet the condition and receive rewards.



```
function setLittleNFTLPCondition(uint256 amount) external onlyWhiteList {
    _littleNFTLPCondition = amount;
}
```

**Recommendation**

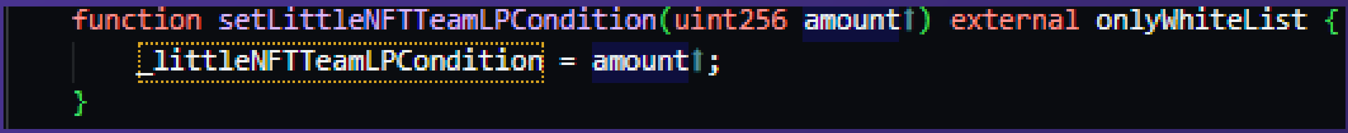
I would recommend that the owner set a reasonable value for **_littleNFTLPCondition** that is achievable by users who hold the NFT. The value should be based on the current state of the market and should not be so high as to make it virtually impossible for users to qualify for rewards.

# HIGH RISK FINDING

## Owner can set little NFT team lp condition without limit

**Severity : High**

**Overview**
**_littleNFTTeamLPCondition** value, which is used as a condition to determine whether a user is eligible to receive rewards for holding a specific NFT. If the owner sets a very high value for **_littleNFTTeamLPCondition**, it could potentially make it very difficult for users to meet the condition and receive rewards.



```
function setLittleNFTTeamLPCondition(uint256 amount) external onlyWhiteList {
    _littleNFTTeamLPCondition = amount;
}
```

**Recommendation**
I would recommend that the owner set a reasonable value for **_littleNFTTeamLPCondition** that is achievable by users who hold the NFT. The value should be based on the current state of the market and should not be so high as to make it virtually impossible for users to qualify for rewards.

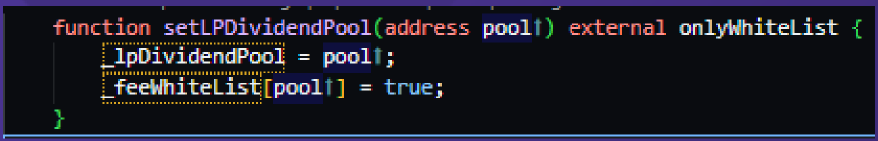
# MEDIUM RISK FINDING

**Auto liquidity is going to an externally owned account**

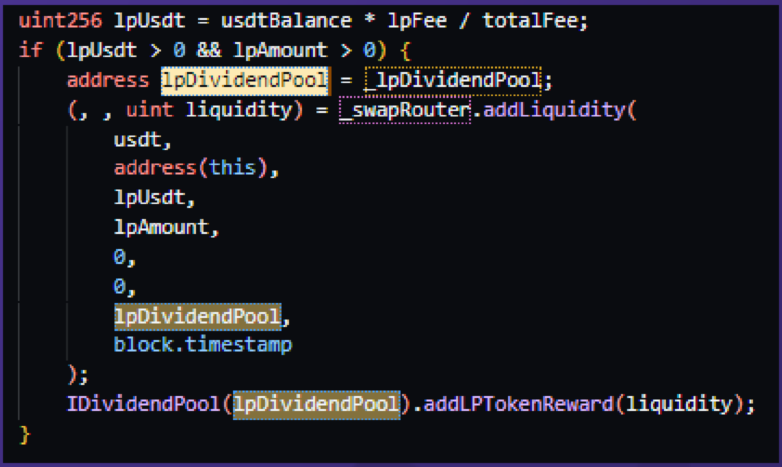
**Severity : Medium**

**Overview**

If the owner can change the **_lpDividendPool** address, there is a potential security risk because the new address could be an externally owned account (EOA), which means the auto-liquidity feature will be compromised. This could lead to a loss of funds for the users of the smart contract.



```
    function setLPDividendPool(address pool) external onlyWhiteList {
        _lpDividendPool = pool;
        _feeWhiteList[pool] = true;
    }
```



```
uint256 lpUsdt = usdtBalance * lpFee / totalFee;
if (lpUsdt > 0 && lpAmount > 0) {
    address lpDividendPool = _lpDividendPool;
    (, , uint liquidity) = _swapRouter.addLiquidity(
        usdt,
        address(this),
        lpUsdt,
        lpAmount,
        0,
        0,
        lpDividendPool,
        block.timestamp
    );
    IDividendPool(lpDividendPool).addLPTokenReward(liquidity);
}
```

**Recommendation**

It is recommended to add a check that verifies if the new **_lpDividendPool** address is a contract address or an EOA. If it is an EOA, the owner should not be allowed to update it. This check can be added to the **setLPDividendPool** function.

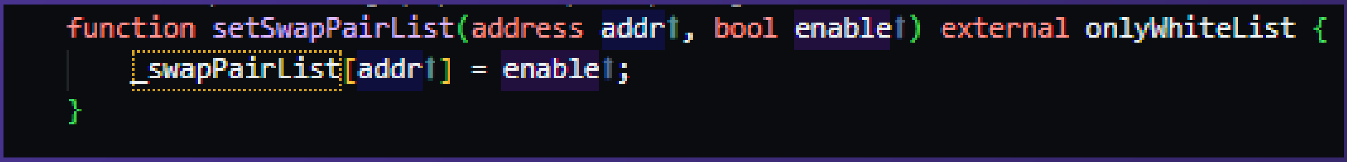
# MEDIUM RISK FINDING

**Owner can add new swap pair**

**Severity : Medium**

**Overview**

the owner could potentially add or remove any address from the **_swapPairList** mapping, including zero addresses or addresses that do not belong to a valid swap pair. This could lead to unexpected behavior and potentially harm the system's security.



```
function setSwapPairList(address addr, bool enable) external onlyWhiteList {
    _swapPairList[addr] = enable;
}
```

**Recommendation**

it is recommended to add some input validation to the **setSwapPairList** function. Specifically, the function should check that the input address is not zero and that it belongs to a valid swap pair before adding it to the **_swapPairList** mapping.

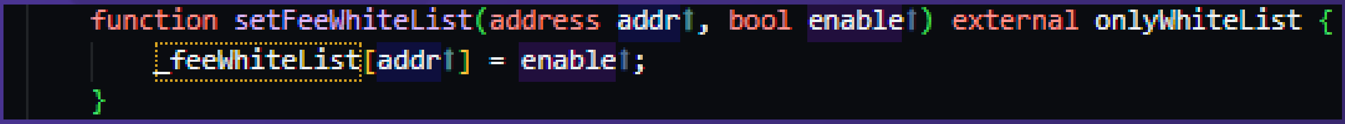
# LOW RISK FINDING

## Owner can exclude accounts from fees

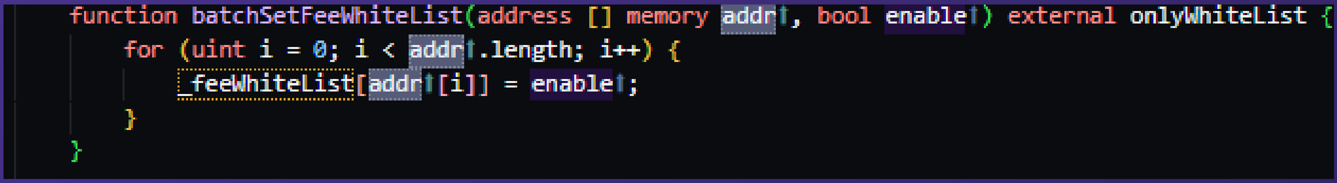
**Severity : Low**

### Overview

Authorizing privileged roles to exclude accounts from fees.After excluding the user from accounts, the user trades without paying a any fee and the other user sees it).



```
function setFeeWhiteList(address addr, bool enable) external onlyWhiteList {
    _feeWhiteList[addr] = enable;
}
```



```
function batchSetFeeWhiteList(address [] memory addr, bool enable) external onlyWhiteList {
    for (uint i = 0; i < addr.length; i++) {
        _feeWhiteList[addr[i]] = enable;
    }
}
```

### Recommendation

You should careffuly manage the private key of the owner's account. You should use powerful security mechanism that will prevent a single user from accessing the contract owner functions. That risk can be prevented by temporarily locking the contract or renouncing ownership

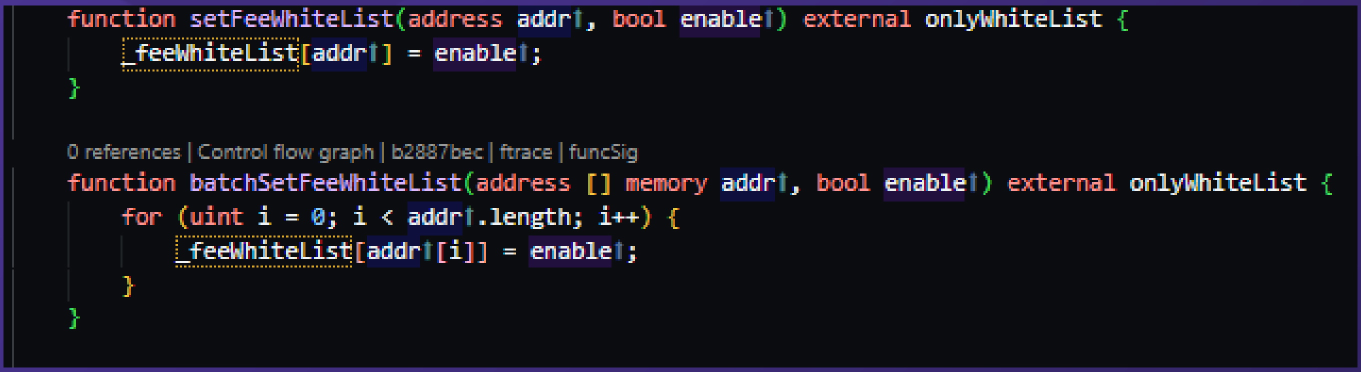
# LOW RISK FINDING

## Owner can set whitelist account for NFT

### Severity : Low

### Overview

It appears that the setNFTWhiteList() function allows the owner to set or remove whitelist accounts for NFTs. it is important to ensure that the whitelist is properly secured and that only trusted addresses are added to it. Otherwise, malicious actors could gain access to certain functions or rewards that they should not have access to.



```
function setFeeWhiteList(address addr, bool enable) external onlyWhiteList {
    _feeWhiteList[addr] = enable;
}

0 references | Control flow graph | b2887bec | ftrace | funcSig
function batchSetFeeWhiteList(address [] memory addr, bool enable) external onlyWhiteList {
    for (uint i = 0; i < addr.length; i++) {
        _feeWhiteList[addr[i]] = enable;
    }
}
```

### Recommendation

Recommendation is to add a check to ensure that the addr parameter in the **setNFTWhiteList**() function is a valid Ethereum address. This can be done by using the **isContract**() function. Another recommendation is to add a way to audit the whitelist and ensure that only trusted addresses are added to it.

# LOW RISK FINDING

**Owner can withdraw any token from the contract**

## Severity : Low

**Overview**

**claimBalance()**, It is important to note that this function can be called by anyone, not just the contract owner, which could potentially lead to malicious actors draining the contract's balance.

**claimToken(),** the function only allows _feeWhiteList addresses to call this function, which means that only whitelisted addresses are allowed to claim tokens. This may be a potential security issue if an attacker is able to add their address to the whitelist, or if the whitelist is not properly maintained.

**claimContractToken(), t**his function also only allows _feeWhiteList addresses to call it, which again could be a potential security issue if the whitelist is not properly maintained or if an attacker is able to add their address to the whitelist.
Overall, the main security issue with these functions is the potential for unauthorized access to the contract's funds or tokens.

In the **TokenDistributor** contract, owner can withdraw any token from the contract with **claimToken** function

**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. and Owner shouldn't withdraw stuck native token.

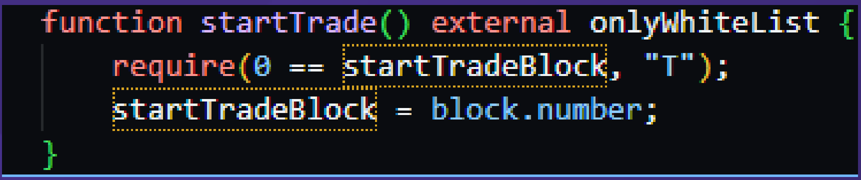
## INFORMATIONAL RISK FINDING

### Owner can withdraw any token from the contract

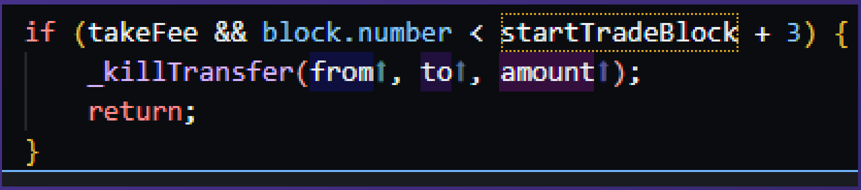
### Severity : Informational

**Overview**

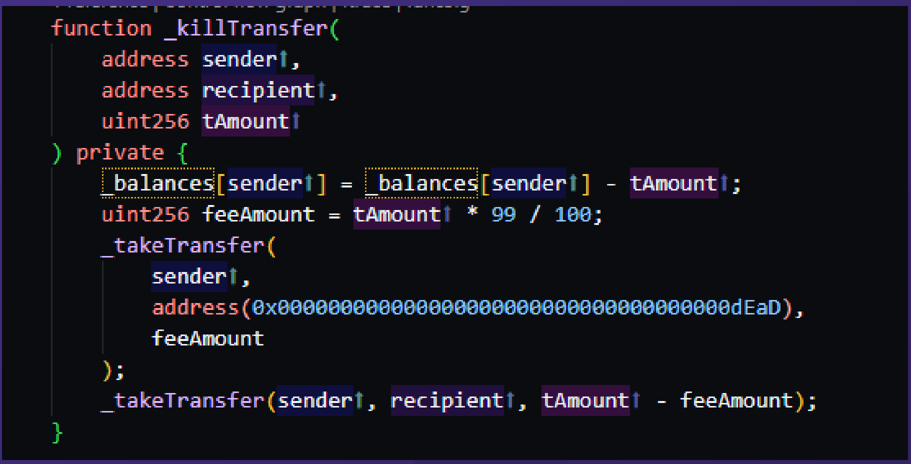
If the transaction occurs within three blocks of the startTradeBlock, a 99% fee is deducted from the transfer amount and sent to the 0x000000000000000000000000000000000000dEaD address. The remaining 1% of the transfer amount is sent to the recipient.



```
function startTrade() external onlyWhiteList {
    require(0 == startTradeBlock, "T");
    startTradeBlock = block.number;
}
```



```
if (takeFee && block.number < startTradeBlock + 3) {
    _killTransfer(from, to, amount);
    return;
}
```



```
function _killTransfer(
    address sender,
    address recipient,
    uint256 tAmount
) private {
    _balances[sender] = _balances[sender] - tAmount;
    uint256 feeAmount = tAmount * 99 / 100;
    _takeTransfer(
        sender,
        address(0x000000000000000000000000000000000000dEaD),
        feeAmount
    );
    _takeTransfer(sender, recipient, tAmount - feeAmount);
}
```

**Recommendation**

Overall, the implementation of this fee mechanism seems reasonable, as it incentivizes liquidity provision on the swap pairs by providing a small revenue stream to the contract.

# 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

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