



Building the Futuristic **Blockchain** Ecosystem

# SECURITY AUDIT REPORT

Myro Floki CEO

# TOKEN OVERVIEW

## Risk Findings

Severity	Found
● High	2
● Medium	0
● Low	2
● Informational	2

## 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

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	Static Analysis	_____
13	Manual Review	_____
21	About Expelee	_____
22	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>Fail</b>
<b>Audit Date</b>	<b>3 March 2024</b>

# CONTRACT DETAILS

**Token Address:** --

**Name:** Myro Floki CEO

**Symbol:** MyroFloki

**Decimals:** 18

**Network:** --

**Token Type:** --

**Owner:** --

**Deployer:** --

**Token Supply:** 10000000000

**Checksum:** A2032c616934aeb47e6039f76b20df21

**Testnet:**

<https://testnet.bscscan.com/address/0xa424552DDa670e2a96D22a8821735343AA3E1294#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 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

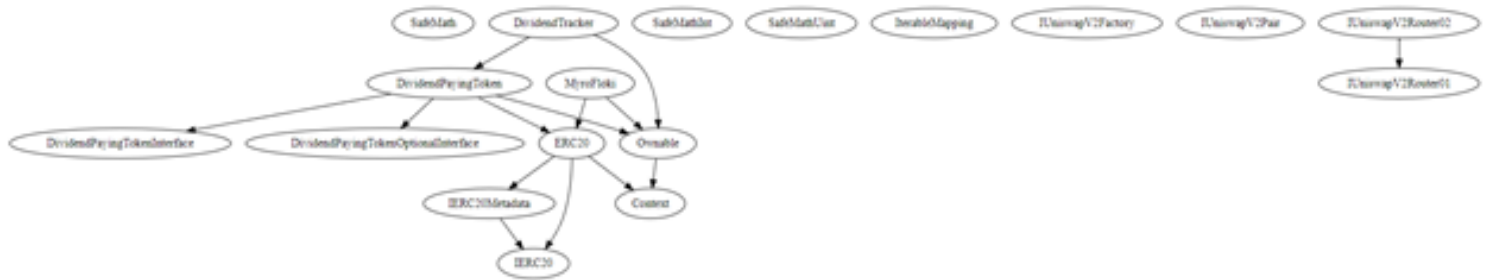
Issues on this level are minor details and warnings that can remain unfixed.

## Informational

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



# INHERITANCE TREES



# STATIC ANALYSIS

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

```
INFO:Detectors:
DividendPayingToken.constructor(string,string,address)._name (MyroFloki.sol#1155) shadows:
- ERC20._name (MyroFloki.sol#489) (state variable)
DividendPayingToken.constructor(string,string,address)._symbol (MyroFloki.sol#1155) shadows:
- ERC20._symbol (MyroFloki.sol#489) (state variable)
DividendPayingToken.dividendOf(address)._owner (MyroFloki.sol#1191) shadows:
- Ownable._owner (MyroFloki.sol#269) (state variable)
DividendPayingToken.withdrawableDividendOf(address)._owner (MyroFloki.sol#1197) shadows:
- Ownable._owner (MyroFloki.sol#269) (state variable)
DividendPayingToken.withdrawnDividendOf(address)._owner (MyroFloki.sol#1281) shadows:
- Ownable._owner (MyroFloki.sol#269) (state variable)
DividendPayingToken.accumulativeDividendOf(address)._owner (MyroFloki.sol#1285) shadows:
- Ownable._owner (MyroFloki.sol#269) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
DividendTracker.setLastProcessedIndex(uint256) (MyroFloki.sol#1299-1381) should emit an event for:
- lastProcessedIndex = index (MyroFloki.sol#1380)
MyroFloki.setSwapTokensAtAmount(uint256) (MyroFloki.sol#1761-1764) should emit an event for:
- swapTokensAtAmount = newAmount (MyroFloki.sol#1763)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
DividendPayingToken.constructor(string,string,address)._rewardToken (MyroFloki.sol#1155) lacks a zero-check on :
- rewardToken = _rewardToken (MyroFloki.sol#1156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
DividendPayingToken._withdrawDividendOfUser(address) (MyroFloki.sol#1176-1191) has external calls inside a loop: success = IERC20(rewardToken).transfer(user._withdrawableDividend) (MyroFloki.sol#1181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#calls-inside-a-loop
INFO:Detectors:
Reentrancy in DividendTracker.processAccount(address,bool) (MyroFloki.sol#1407-1457):
External calls:
- amount = _withdrawDividendOfUser(account) (MyroFloki.sol#1448)
- success = IERC20(rewardToken).transfer(user._withdrawableDividend) (MyroFloki.sol#1181)
State variables written after the call(s):
- lastClaimTimes[account] = block.timestamp (MyroFloki.sol#1451)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Reentrancy in MyroFloki._setAutomatedMarketMakerPair(address,bool) (MyroFloki.sol#1577-1586):
External calls:
- dividendTracker.excludeFromDividends(pair) (MyroFloki.sol#1582)
Event emitted after the call(s):
- SetAutomatedMarketMakerPair(pair,value) (MyroFloki.sol#1585)
Reentrancy in MyroFloki._transfer(address,address,uint256) (MyroFloki.sol#1641-1738):
External calls:
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(swapAmount,0,path,address(this),block.timestamp) (MyroFloki.sol#1677-1683)
- (success) = marketingWallet.call{value: marketingShare}() (MyroFloki.sol#1689)
External calls sending eth:
- (success) = marketingWallet.call{value: marketingShare}() (MyroFloki.sol#1689)
Event emitted after the call(s):
- TransferFailed(marketingWallet,marketingShare) (MyroFloki.sol#1691)
Reentrancy in MyroFloki._transfer(address,address,uint256) (MyroFloki.sol#1641-1738):
External calls:
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(swapAmount,0,path,address(this),block.timestamp) (MyroFloki.sol#1677-1683)
- (success) = marketingWallet.call{value: marketingShare}() (MyroFloki.sol#1689)
- swapAndSendDividends(address(this).balance - initialBalance) (MyroFloki.sol#1697)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,address(this),block.timestamp) (MyroFloki.sol#1745-1750)
- success = IERC20(rewardToken).transfer(address(dividendTracker),balanceRewardToken) (MyroFloki.sol#1753)
- dividendTracker.distributeDividends(balanceRewardToken) (MyroFloki.sol#1756)
External calls sending eth:
- (success) = marketingWallet.call{value: marketingShare}() (MyroFloki.sol#1689)
- swapAndSendDividends(address(this).balance - initialBalance) (MyroFloki.sol#1697)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,address(this),block.timestamp) (MyroFloki.sol#1745-1750)
Event emitted after the call(s):
- SendDividends(balanceRewardToken) (MyroFloki.sol#1757)
- swapAndSendDividends(address(this).balance - initialBalance) (MyroFloki.sol#1697)
- Transfer(from,to,amount) (MyroFloki.sol#681)
- super._transfer(from,address(this),fees) (MyroFloki.sol#1721)
- Transfer(from,to,amount) (MyroFloki.sol#681)
- super._transfer(from,to,amount) (MyroFloki.sol#1724)
Reentrancy in MyroFloki._transfer(address,address,uint256) (MyroFloki.sol#1641-1738):
External calls:
- uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(swapAmount,0,path,address(this),block.timestamp) (MyroFloki.sol#1677-1683)
- (success) = marketingWallet.call{value: marketingShare}() (MyroFloki.sol#1689)
- swapAndSendDividends(address(this).balance - initialBalance) (MyroFloki.sol#1697)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,address(this),block.timestamp) (MyroFloki.sol#1745-1750)
- success = IERC20(rewardToken).transfer(address(dividendTracker),balanceRewardToken) (MyroFloki.sol#1753)
- dividendTracker.distributeDividends(balanceRewardToken) (MyroFloki.sol#1756)
- dividendTracker.setBalance(address(from),balanceOf(from)) (MyroFloki.sol#1726)
- dividendTracker.setBalance(address(to),balanceOf(to)) (MyroFloki.sol#1727)
- (iterations,claims,lastProcessedIndex) = dividendTracker.process(gas) (MyroFloki.sol#1732-1736)
External calls sending eth:
- (success) = marketingWallet.call{value: marketingShare}() (MyroFloki.sol#1689)
- swapAndSendDividends(address(this).balance - initialBalance) (MyroFloki.sol#1697)
- uniswapV2Router.swapExactETHForTokensSupportingFeeOnTransferTokens{value: amount}(0,path,address(this),block.timestamp) (MyroFloki.sol#1745-1750)
Event emitted after the call(s):
```

# STATIC ANALYSIS

```
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (MyroFloki.sol#948) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (MyroFloki.sol#949) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (MyroFloki.sol#966) is not in mixedCase
Function IUniswapV2Router01.WETH() (MyroFloki.sol#986) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (MyroFloki.sol#1193) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (MyroFloki.sol#1197) is not in mixedCase
Parameter DividendPayingToken.withdrawDividendOf(address)._owner (MyroFloki.sol#1201) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (MyroFloki.sol#1205) is not in mixedCase
Constant DividendPayingToken.magnitude (MyroFloki.sol#1146) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.updateMinimumTokenBalanceForDividends(uint256)._newMinimumBalance (MyroFloki.sol#1277) is not in mixedCase
Parameter DividendTracker.getAccount(address)._account (MyroFloki.sol#1311) is not in mixedCase
Event MyroFloki.marketingWalletChanged(address) (MyroFloki.sol#1489) is not in CamelCase
Parameter MyroFloki.withdrawTokenTokens(address).REF28_token (MyroFloki.sol#1566) is not in mixedCase
Parameter MyroFloki.updateBuyFees(uint256,uint256)._marketingFeeOnBuy (MyroFloki.sol#1587) is not in mixedCase
Parameter MyroFloki.updateBuyFees(uint256,uint256)._rewardsFeeOnBuy (MyroFloki.sol#1587) is not in mixedCase
Parameter MyroFloki.updateSellFees(uint256,uint256)._marketingFeeOnSell (MyroFloki.sol#1608) is not in mixedCase
Parameter MyroFloki.updateSellFees(uint256,uint256)._rewardsFeeOnSell (MyroFloki.sol#1608) is not in mixedCase
Parameter MyroFloki.setWhitelistStatus(address,bool)._wallet (MyroFloki.sol#1611) is not in mixedCase
Parameter MyroFloki.setWhitelistStatus(address,bool)._status (MyroFloki.sol#1612) is not in mixedCase
Parameter MyroFloki.setmarketingWallet(address)._newmarketing (MyroFloki.sol#1622) is not in mixedCase
Constant MyroFloki._totalSupply (MyroFloki.sol#1475) is not in UPPER_CASE_WITH_UNDERSCORES
Constant MyroFloki.maxSwapAmount (MyroFloki.sol#1479) is not in UPPER_CASE_WITH_UNDERSCORES
Reference: https://github.com/cryptic/sliether/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (MyroFloki.sol#991) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (MyroFloki.sol#992)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (MyroFloki.sol#1177) is too similar to DividendTracker.getAccount(address).withdrawableDividends (MyroFloki.sol#1316)
Reference: https://github.com/cryptic/sliether/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
SafeMathInt.MAX_INT256 (MyroFloki.sol#8217) is never used in SafeMathInt (MyroFloki.sol#8215-8832)
Reference: https://github.com/cryptic/sliether/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
MyroFloki.dividendTracker (MyroFloki.sol#1608) should be immutable
MyroFloki.uniswapV2Pair (MyroFloki.sol#1471) should be immutable
MyroFloki.uniswapV2Router (MyroFloki.sol#1478) should be immutable
Reference: https://github.com/cryptic/sliether/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Sliether:MyroFloki.sol analyzed (18 contracts with 93 detectors), 83 result(s) found
```

# TESTNET VERSION

## 1- Approve (passed):

<https://testnet.bscscan.com/tx/0xa3f78caf284980bddec8e92105f51728f92cfe5694103c0aa621b2d510cc21fb>

## 2- Increase Allowance (passed):

<https://testnet.bscscan.com/tx/0x4f4a53b3a54fcc4ec936d5d93a15b374c069c4faa77c6b2b2b5f3871f22b2b04>

## 3- Decrease Allowance (passed):

<https://testnet.bscscan.com/tx/0xe5598f689fde6e29758138c8e1472dae5db1bc586a598b3e240c767bba297d06>

## 4- Enable Trading (passed):

<https://testnet.bscscan.com/tx/0xf176e62789fd1ba3b18fa6dde6ae47c4eed0709c65f340aa30a07346ff543c8e>

## 5- Update Buy Taxes (passed):

<https://testnet.bscscan.com/tx/0x44ebf7da415bc81a2949213e7afcb02940cb4e65ac4969bf2ce5b407932e291a>

## 6- Update Sell Taxes (passed):

<https://testnet.bscscan.com/tx/0x28de5f479e63f1c7fcb2b9cb8bb0c54e191381e26293cf14243bd7fe1974b1fb>

## 7- Transfer (passed):

<https://testnet.bscscan.com/tx/0x0daefbff8ac4b59981e79715ac449757f525d27da8bddcf4187ec1c7a66b2b19>

## 8- Set marketing Wallet (passed):

<https://testnet.bscscan.com/tx/0xca3478f6053b7be9aa3878ac5d86983c57dd7c6df3d80c64514cafee52f8bbc8>

# 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			

# HIGH RISK FINDING

**Centralization** – Missing Require Check

Severity: **High**

function: SetMarketingWallet

Status: Open

## Overview:

The owner can set any arbitrary address excluding zero address as this is not recommended because if the owner sets the address to the contract address, then the ETH will not be sent to that address and the transaction will fail and this will lead to a potential honeypot in the contract.

```
function setmarketingWallet(address _newmarketing) external  
onlyOwner {  
    require(  
        _newmarketing != address(0),  
        "can not set marketing to dead wallet"  
    );  
    require(!isContract(_newmarketing), "Address can not be a  
contract address");  
    marketingWallet = _newmarketing;  
    emit marketingWalletChanged(_newmarketing);  
}
```

## Suggestion:

It is recommended that the address should not be able to set as a contract address.



# HIGH RISK FINDING

## Centralization – Enabling Trades

Severity: High

function: EnableTrading

Status: Open

### Overview:

The EnableTrading 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 enableTrading() external onlyOwner {  
    require(!tradingEnabled, "Trading is already enabled");  
    tradingEnabled = true;  
    startTradingBlock = 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.

# 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 setLastProcessedIndex(uint256 index) external  
onlyOwner {  
    dividendTracker.setLastProcessedIndex(index);  
}
```

### Suggestion

Emit an event for critical changes.



# LOW RISK FINDING

## Centralization – Unsafe Usage of tx.origin

Severity: Low

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.

```
function processDividendTracker(uint256 gas) external {  
    (uint256 iterations, uint256 claims, uint256 lastProcessedIndex)  
    = dividendTracker.process(gas);  
    emit ProcessedDividendTracker(iterations, claims,  
    lastProcessedIndex, false, gas, tx.origin);  
}
```

### 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).

# INFORMATIONAL & OPTIMIZATIONS

## Optimization

Severity: **Informational**

subject: **Remove Safe Math**

Status: **Open**

Line: **20–219**

### 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: 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 virtual returns (bytes  
calldata) {  
    return msg.data;  
}
```

```
function mul(int256 a, int256 b) internal pure returns (int256) {  
    int256 c = a * b;
```

```
// Detect overflow when multiplying MIN_INT256 with -1  
require(c != MIN_INT256 || (a & MIN_INT256) != (b &  
MIN_INT256));  
require((b == 0) || (c / b == a));  
return c;  
}
```

# INFORMATIONAL & OPTIMIZATIONS

```
function div(int256 a, int256 b) internal pure returns (int256) {  
    // Prevent overflow when dividing MIN_INT256 by -1  
    require(b != -1 || a != MIN_INT256);
```

```
    // Solidity already throws when dividing by 0.  
    return a / b;  
}
```

```
function abs(int256 a) internal pure returns (int256) {  
    require(a != MIN_INT256);  
    return a < 0 ? -a : a;  
}
```

```
event SwapAndLiquify(uint256 tokensSwapped, uint256  
    bnbReceived, uint256 tokensIntoLiquidity);  
event UpdateUniswapV2Router(address indexed newAddress,  
    address indexed oldAddress);  
event UpdateDividendTracker(address indexed newAddress,  
    address indexed oldAddress);
```

# 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/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 protect 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. Always 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**