expelee

Building the Futuristic Blockchain Ecosystem

# SECURITY AUDIT REPORT

MPEP

# 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:

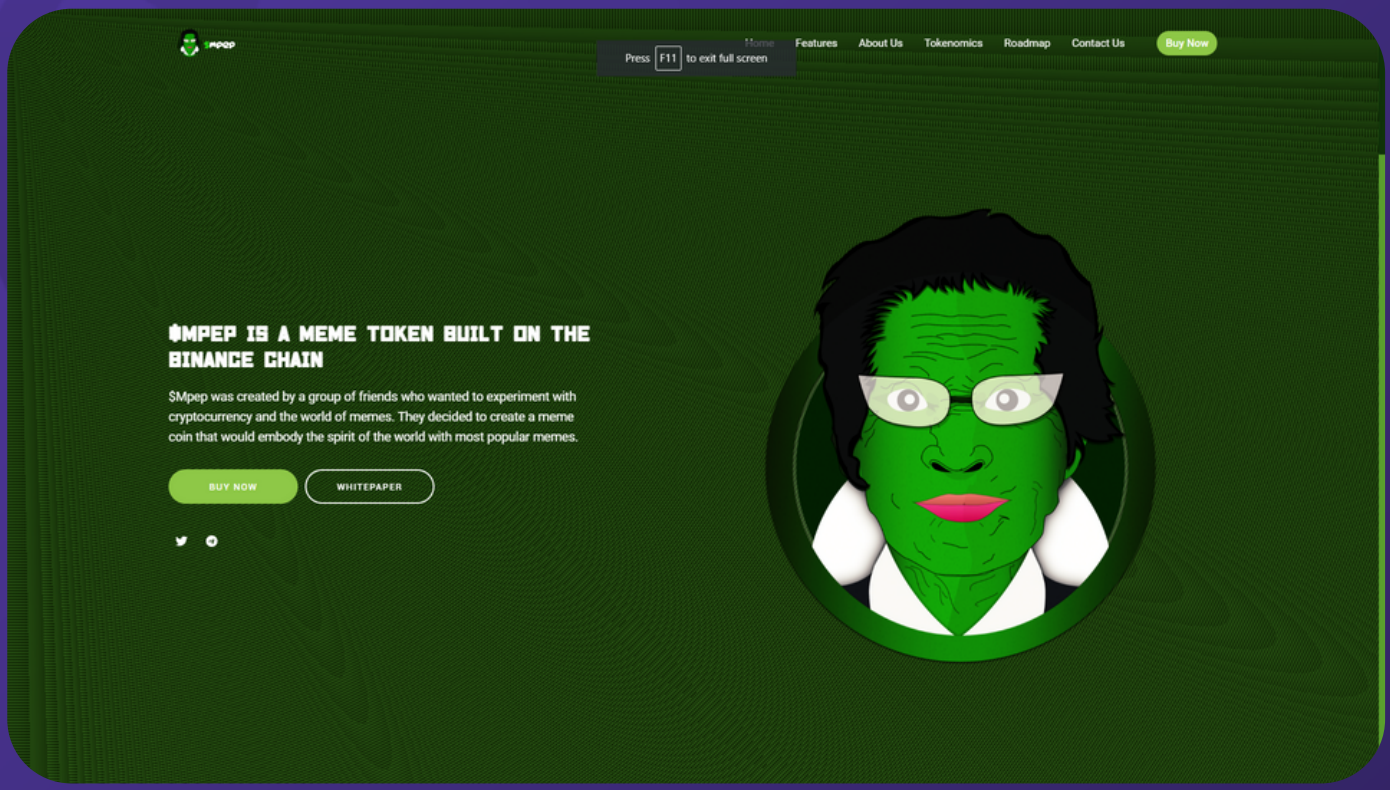
| | |
|---|---|
| **Audit Result** | **Passed** |
| **KYC Verification** | **-** |
| **Audit Date** | **22 May 2023** |

# PROJECT DESCRIPTION

$Mpep was created by a group of friends who wanted to experiment with cryptocurrency and the world of memes. They decided to create a meme coin that would embody the spirit of the world with most popular memes.

# SOCIAL MEDIA PROFILES

## MPEP




https://t.me/Mpepbsc

https://twitter.com/MpepOfficial1?s=20

https://mpepbsc.com/

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

Team Expelee

# CONTRACT DETAILS

Token Name: MPEP

Symbol: MPEP

Network: Binance Smart Chain

Language: Solidity

Contract Address:
0x8bccab8C45909aAf850eA4890701a54308729B31

Total Supply: 1000000000000

Owner's Wallet:
0x90106Fc2c22c0b895Ab2808c32414f700D21E391

Deployer's Wallet:
0x90106Fc2c22c0b895Ab2808c32414f700D21E391

Testnet:
https://testnet.bscscan.com/address/0x0f5931DA51FBE4Fe881dB6458a4D0566009EfCbB

# OWNER PRIVILEGES

- Owner can exclude accounts from rewards
- Owner can exclude accounts from fees
- The owner can set max transaction amount within reasonable limits
- Owner can change swap token at amount within reasonable limits
- Owner can update Router address
- Owner can update Automated market maker pair
- Owner can update Dividend tracker address
- Owner can change claimWait
- Owner can change LiquidityWallet

# 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.

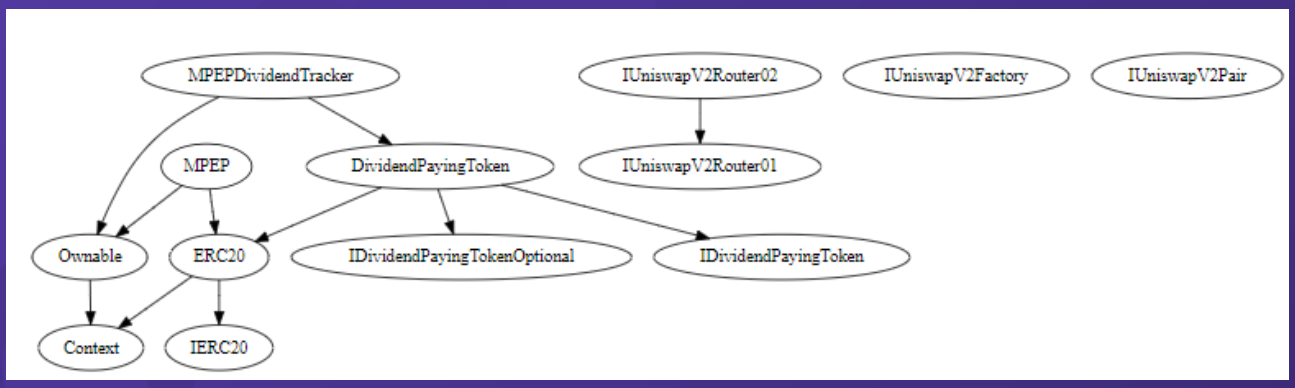
## Low Risk

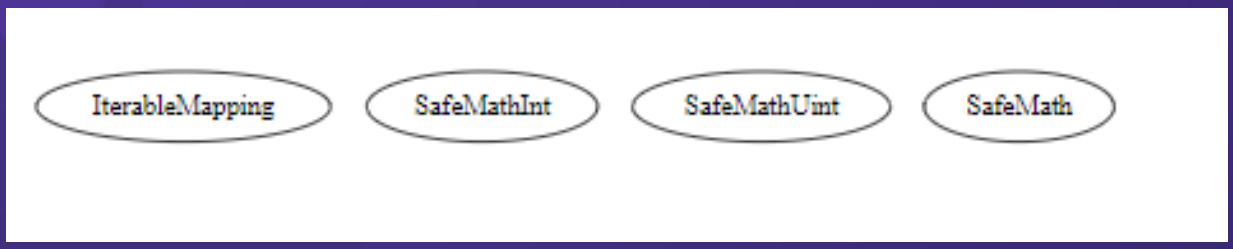
Issues on this level are minor details and warning that can remain unfixed.

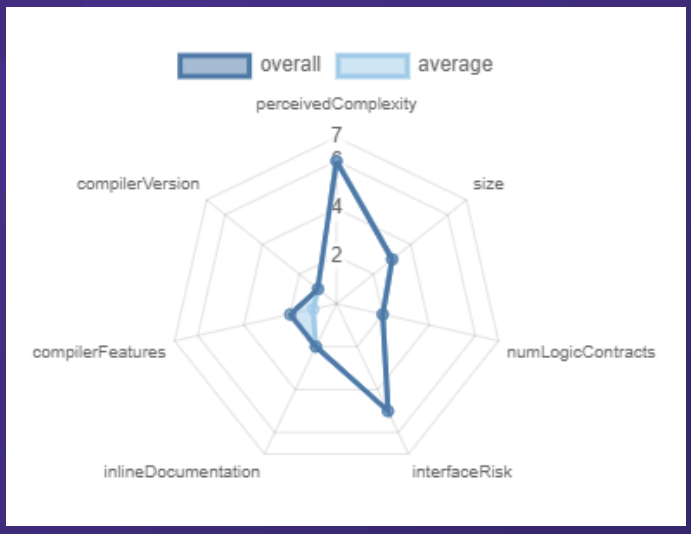
## Informational

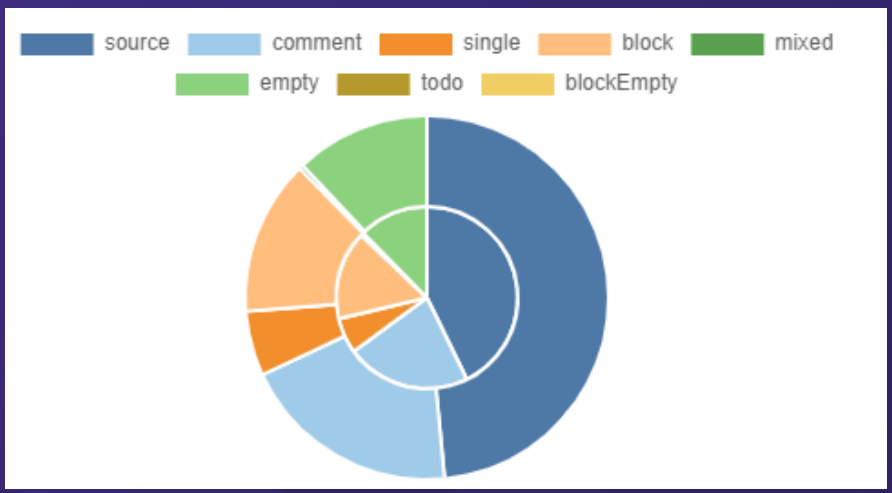
Issues on this level are minor details and warning that can remain unfixed.

# INHERITANCE TREES



expelee

# FUNCTION DETAILS

| | | | | |
|---|---|---|---|---|
| **MPEP** | Implementation | ERC20, Ownable | | |
| └ | <Constructor> | Public ❗ | 🛑 | ERC20 |
| └ | <Receive Ether> | External ❗ | 💵 | NO ❗ |
| └ | updateDividendTracker | Public ❗ | 🛑 | onlyOwner |
| └ | updateUniswapV2Router | Public ❗ | 🛑 | onlyOwner |
| └ | excludeFromFees | Public ❗ | 🛑 | onlyOwner |
| └ | excludeMultipleAccountsFromFees | Public ❗ | 🛑 | onlyOwner |
| └ | setAutomatedMarketMakerPair | Public ❗ | 🛑 | onlyOwner |
| └ | _setAutomatedMarketMakerPair | Private 🔐 | 🛑 | |
| └ | updateLiquidityWallet | Public ❗ | 🛑 | onlyOwner |
| └ | updateGasForProcessing | Public ❗ | 🛑 | onlyOwner |
| └ | updateClaimWait | External ❗ | 🛑 | onlyOwner |
| └ | getClaimWait | External ❗ | | NO ❗ |
| └ | getTotalDividendsDistributed | External ❗ | | NO ❗ |
| └ | SetmaxSellTransactionAmount | External ❗ | 🛑 | onlyOwner |
| └ | setswapTokensAtAmount | External ❗ | 🛑 | onlyOwner |
| └ | isExcludedFromFees | Public ❗ | | NO ❗ |
| └ | withdrawableDividendOf | Public ❗ | | NO ❗ |
| └ | dividendTokenBalanceOf | Public ❗ | | NO ❗ |
| └ | getAccountDividendsInfo | External ❗ | | NO ❗ |
| └ | getAccountDividendsInfoAtIndex | External ❗ | | NO ❗ |
| └ | processDividendTracker | External ❗ | 🛑 | NO ❗ |
| └ | claim | External ❗ | 🛑 | NO ❗ |
| └ | getLastProcessedIndex | External ❗ | | NO ❗ |
| └ | getNumberOfDividendTokenHolders | External ❗ | | NO ❗ |
| └ | getTradingIsEnabled | Public ❗ | | NO ❗ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | swapAndLiquify | Private 🔐 | 🛑 | |
| └ | swapTokensForEth | Private 🔐 | 🛑 | |
| └ | swapTokensForBNB | Private 🔐 | 🛑 | |
| └ | addLiquidity | Private 🔐 | 🛑 | |
| └ | swapAndSendDividends | Private 🔐 | 🛑 | |
| **MPEPDividendTracker** | Implementation | DividendPayingToken, Ownable | | |
| └ | <Constructor> | Public ❗ | 🛑 | DividendPayingToken |
| └ | _transfer | Internal 🔒 | | |
| └ | withdrawDividend | Public ❗ | | NO ❗ |
| └ | excludeFromDividends | External ❗ | 🛑 | onlyOwner |
| └ | updateClaimWait | External ❗ | 🛑 | onlyOwner |
| └ | getLastProcessedIndex | External ❗ | | NO ❗ |
| └ | getNumberOfTokenHolders | External ❗ | | NO ❗ |
| └ | getAccount | Public ❗ | | NO ❗ |
| └ | getAccountAtIndex | Public ❗ | | NO ❗ |
| └ | canAutoClaim | Private 🔐 | | |
| └ | setBalance | External ❗ | 🛑 | onlyOwner |
| └ | process | Public ❗ | 🛑 | NO ❗ |
| └ | processAccount | Public ❗ | 🛑 | 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 | | | |

expelee

# FINDINGS

| Findings | Severity | Found |
|---|---|---|
| High Risk | High | 0 |
| Medium Risk | Medium | 1 |
| Low Risk | Low | 3 |
| Suggestion & discussion | Informational | 0 |
| Gas Optimizations | Gas Opt. | 0 |

# MEDIUM RISK FINDING

## Owner can exclude accounts from rewards

### Severity : Medium

**Overview**

Function that allows the owner of the contract to exclude an address from receiving dividends

```
function excludeFromDividends(address account) external onlyOwner {
    require(!excludedFromDividends[account]);
    excludedFromDividends[account] = true;

    _setBalance(account, 0);
    tokenHoldersMap.remove(account);

    emit ExcludeFromDividends(account);
}
```

**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. The contract owner account is well secured and only accessible by authorized parties.

# LOW RISK FINDING

## Owner can exclude accounts from fees

## Severity : Low

### Overview

Excludes/Includes an address from the collection of fees

```
3 references | Control flow graph | c024b6b8 | ftrace | funcSig
function excludeFromFees(address account, bool excluded) public onlyOwner {
    require(
        _isExcludedFromFees[account] != excluded,
        "MPEP: Account is already the value of 'excluded'"
    );
    _isExcludedFromFees[account] = excluded;

    emit ExcludeFromFees(account, excluded);
}

0 references | Control flow graph | c492f046 | ftrace | funcSig
function excludeMultipleAccountsFromFees(
    address[] calldata accounts,
    bool excluded
) public onlyOwner {
    for (uint256 i = 0; i < accounts.length; i++) {
        _isExcludedFromFees[accounts[i]] = excluded;
    }

    emit ExcludeMultipleAccountsFromFees(accounts, excluded);
}
```

### 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. The contract owner account is well secured and only accessible by authorized parties.

# LOW RISK FINDING

**The owner can set max transaction amount within reasonable limits**

## Severity : Low

### Overview

**SetmaxSellTransactionAmount** function that allows the contract owner to set the maximum sell tx amount

```
function SetmaxSellTransactionAmount(uint256 maxSellTransactionAmount)
    external
    onlyOwner
{
    require(
        maxSellTransactionAmount >= 1000000000000 * (10**18),
        "SellFee cant be lower than 1000000000000"
    );
    _maxSellTransactionAmount = maxSellTransactionAmount;
}
```

### Recommendation

Verify that appropriate access control mechanisms are in place to restrict the **_maxSellTransactionAmount** function to the contract owner only. Ensure that the **onlyOwner** modifier is correctly implemented and that ownership cannot be easily transferred or compromised.

# LOW RISK FINDING

## Owner can change swap token at amount within reasonable limits

## Severity : Low

### Overview

setSwapTokensAtAmount function allows the owner of the contract to update the value of swapTokensAtAmount.

```
function setswapTokensAtAmount(uint256 swapTokensAtAmount)
    external
    onlyOwner
{
    uint256 contractTokenBalance = balanceOf(address(this));
    require (swapTokensAtAmount > 1000000 * (10**18),
    "swapTokensAtAmount cant be lower than 1000000"
    );
    require (swapTokensAtAmount < 4000000 * (10**18),
    "swapTokensAtAmount cant be more than 4000000"
    );
    require(
        swapTokensAtAmount >= contractTokenBalance,
        "swapTokensAtAmount cant be lower than current contract acumullated balance"
    );
    swapTokensAtAmount = swapTokensAtAmount;
}
```

### Recommendation

IIf the threshold is set too low, it could result in frequent and unnecessary swaps, which would increase gas fees and potentially lead to losses due to slippage. On the other hand, if the threshold is set too high, it could result in liquidity being insufficient to handle large trades, which could negatively impact the token price and liquidity pool. Be ensure that the contract owner account is well secured and only accessible by authorized parties.

# 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.