



Building the Futuristic **Blockchain Ecosystem**

SECURITY AUDIT REPORT

PAJEET

TOKEN OVERVIEW

Risk Findings

Severity	Found
● High	1
● Medium	0
● Low	1
● Informational	2

Centralization Risks

Owner Privileges	Description
● Can Owner Set Taxes >25% ?	Not Detected
● Owner needs to enable trading ?	Not Detected
● 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	
11	Testnet Version	
12	Manual Review	
17	About Expelee	
18	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:

Audit Result	Passed with high risk
Audit Date	21 March 2024

CONTRACT DETAILS

Token Address: 0xC23851686c6dB2F0A043c7314b46238BD4771f36

Name: Pajeet

Symbol: PAJEET

Decimals: 9

Network: BSC

Token Type: BEP-20

Owner: 0x327166e33F1a1C3874C3a29Ac26FC96b3085C5FE

Deployer: 0x327166e33F1a1C3874C3a29Ac26FC96b3085C5FE

Token Supply: 1000000000

Checksum: A2032c616934aeb47e6039f76b20d221

Testnet:

<https://testnet.bscscan.com/address/0xea7031e54cdb92615f94d4a78ab32ec6ddf18220#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:
PAJEET._transfer(address,address,uint256).burnTokens (PAJEET.sol#1056) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
PAJEET.getAccountDividendsInfo(address) (PAJEET.sol#1195-1206) ignores return value by dividendTracker.getAccount(account) (PAJEET.sol#1205)
PAJEET.getAccountDividendsInfoAtIndex(uint256) (PAJEET.sol#1208-1219) ignores return value by dividendTracker.getAccountAtIndex(index) (PAJEET.sol#1218)
PAJEET.claim() (PAJEET.sol#1226-1228) ignores return value by dividendTracker.processAccount(address(msg.sender),false) (PAJEET.sol#1227)
PAJEET.claimAddress(address) (PAJEET.sol#1230-1232) ignores return value by dividendTracker.processAccount(address(claimee),false) (PAJEET.sol#1231)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
DividendPayingToken.constructor(string,string,address)._name (PAJEET.sol#582) shadows:
- ERC20._name (PAJEET.sol#439) (state variable)
DividendPayingToken.constructor(string,string,address)._symbol (PAJEET.sol#582) shadows:
- ERC20._symbol (PAJEET.sol#448) (state variable)
DividendPayingToken.dividendOf(address)._owner (PAJEET.sol#620) shadows:
- Ownable._owner (PAJEET.sol#24) (state variable)
DividendPayingToken.withdrawableDividendOf(address)._owner (PAJEET.sol#624) shadows:
- Ownable._owner (PAJEET.sol#24) (state variable)
DividendPayingToken.withdrawnDividendOf(address)._owner (PAJEET.sol#628) shadows:
- Ownable._owner (PAJEET.sol#24) (state variable)
DividendPayingToken.accumulativeDividendOf(address)._owner (PAJEET.sol#632) shadows:
- Ownable._owner (PAJEET.sol#24) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
DividendTracker.setLastProcessedIndex(uint256) (PAJEET.sol#726-728) should emit an event for:
- lastProcessedIndex = index (PAJEET.sol#727)
PAJEET.setSwapTokensAtAmount(uint256) (PAJEET.sol#1161-1164) should emit an event for:
- swapTokensAtAmount = newAmount (PAJEET.sol#1163)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
DividendPayingToken.constructor(string,string,address)._rewardToken (PAJEET.sol#582) lacks a zero-check on :
- rewardToken = _rewardToken (PAJEET.sol#583)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
DividendPayingToken._withdrawDividendOfUser(address) (PAJEET.sol#603-618) has external calls inside a loop: success = IERC20(rewardToken).transfer(user,_withdrawableDividend) (PAJEET.sol#608)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (PAJEET.sol#242) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (PAJEET.sol#243) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (PAJEET.sol#260) is not in mixedCase
Function IUniswapV2Router01.WETH() (PAJEET.sol#280) is not in mixedCase
Parameter DividendPayingToken.dividendOf(address)._owner (PAJEET.sol#620) is not in mixedCase
Parameter DividendPayingToken.withdrawableDividendOf(address)._owner (PAJEET.sol#624) is not in mixedCase
Parameter DividendPayingToken.withdrawnDividendOf(address)._owner (PAJEET.sol#628) is not in mixedCase
Parameter DividendPayingToken.accumulativeDividendOf(address)._owner (PAJEET.sol#632) is not in mixedCase
Constant DividendPayingToken.magnitude (PAJEET.sol#573) is not in UPPER_CASE_WITH_UNDERSCORES
Parameter DividendTracker.updateMinimumTokenBalanceForDividends(uint256)._newMinimumBalance (PAJEET.sol#704) is not in mixedCase
Parameter DividendTracker.getAccount(address)._account (PAJEET.sol#738) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (PAJEET.sol#285) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (PAJEET.sol#286)
Variable DividendPayingToken._withdrawDividendOfUser(address)._withdrawableDividend (PAJEET.sol#604) is too similar to DividendTracker.getAccount(address)._withdrawableDividends (PAJEET.sol#743)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
SafeMathInt.MAX_INT256 (PAJEET.sol#111) is never used in SafeMathInt (PAJEET.sol#109-146)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable
INFO:Detectors:
PAJEET.gasForProcessing (PAJEET.sol#915) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
PAJEET.burnFeeOnBuy (PAJEET.sol#888) should be immutable
PAJEET.burnFeeOnSell (PAJEET.sol#894) should be immutable
PAJEET.dividendTracker (PAJEET.sol#913) should be immutable
PAJEET.marketingFeeOnBuy (PAJEET.sol#889) should be immutable
PAJEET.marketingFeeOnSell (PAJEET.sol#895) should be immutable
PAJEET.rewardsFeeOnBuy (PAJEET.sol#890) should be immutable
PAJEET.rewardsFeeOnSell (PAJEET.sol#896) should be immutable
PAJEET.totalBuyFee (PAJEET.sol#892) should be immutable
PAJEET.totalSellFee (PAJEET.sol#898) should be immutable
PAJEET.uniswapV2Pair (PAJEET.sol#903) should be immutable
PAJEET.uniswapV2Router (PAJEET.sol#902) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:PAJEET.sol analyzed (18 contracts with 93 detectors), 70 result(s) found
```

TESTNET VERSION

1- Approve (passed):

<https://testnet.bscscan.com/tx/0x41d17685faf719067f19ac09b96f8f1489f1d5ddca3515ea4dbbbe0ce0dec5a5>

2- Exclude From Dividends (passed):

<https://testnet.bscscan.com/tx/0x5a85493a6b9c5e385f1386ec512743b064552a0a5c7226b87e8fa65faf1dddfc>

3- Exclude From Fees (passed):

<https://testnet.bscscan.com/tx/0xe1d3a9395a71c68a3f44ec0bec4b29d31871fe9d1c89bcb30947ab49b628bf58>

4- Update Marketing Wallet (passed):

<https://testnet.bscscan.com/tx/0x61e918112f9b4282fb0d89e5eda52fdc47c21c1f504af846623963abe456e726>

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: : Update Marketing Wallet

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 updateMarketingWallet(address newWallet) external  
onlyOwner {  
    require(newWallet != address(0), "Fee Address cannot be zero  
address");  
    marketingWallet = newWallet;  
}
```

Suggestion:

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

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 setSwapTokensAtAmount(uint256 newAmount)
external onlyOwner{
require(newAmount > totalSupply() / 100_000,
"SwapTokensAtAmount must be greater than 0.001% of total
supply");
swapTokensAtAmount = newAmount;
}
```

Suggestion:

Emit an event for critical changes.

INFORMATIONAL & OPTIMIZATIONS

Optimization

Severity: **Informational**

Subject: Remove Safe Math

Status: Open

Line: 57–107

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;  
}  
event UpdateUniswapV2Router(address indexed newAddress,  
address indexed oldAddress);  
event UpdateDividendTracker(address indexed newAddress,  
address indexed oldAddress);  
event GasForProcessingUpdated(uint256 indexed newValue,  
uint256 indexed oldValue);  
function isContract(address account) internal view returns (bool) {  
return account.code.length > 0;  
}
```

Suggestion:

To reduce high gas fees. It is suggested to remove unused code from 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](https://twitter.com/expeleeofficial)

 [expelee](https://medium.com/expelee)

 [Expelee](https://t.me/Expelee)

 [expelee](https://in.linkedin.com/company/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 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.

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**