expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Trump Pepe Agenda

TOKEN OVERVIEW

Risk Findings

Severity	Found
🔴 High	1
🟠 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

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	**15 March 2024**

CONTRACT DETAILS

Token Address: 0x5fb7F004657C89Ae98eD6B11AA0C96a9C9b2e41a

Name: Trump Pepe Agenda

Symbol: TRUMPPEPE

Decimals: 18

Network: BSC

Token Type:BEP-20

Owner: 0xDf1Da5D4131085D91d20D1d3AE0a464b6c561980

Deployer: 0xbdAdb16A0FC6dc8B129fe4C256fD7774FC8757c5

Token Supply: 100000000000

Checksum: A2032c616934aeb47e6039f76b20d241

Testnet:
https://testnet.bscscan.com/address/0x69a3ca7018310feeda5e25ac5252733602d25ac3#code

expelee

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.
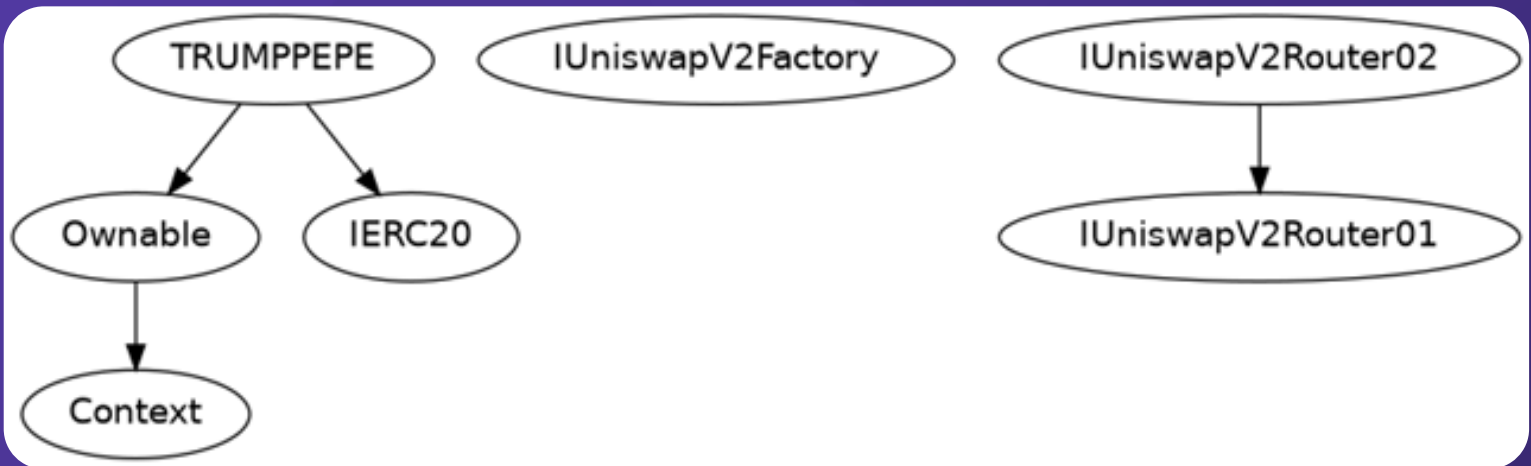
Low Risk

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

STATIC ANALYSIS

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

```
INFO:Detectors:
TRUMPPEPE.constructor(address)._owner (TRUMPPEPE.sol#406) shadows:
        - Ownable._owner (TRUMPPEPE.sol#49) (state variable)
TRUMPPEPE._approve(address,address,uint256).owner (TRUMPPEPE.sol#457) shadows:
        - Ownable.owner() (TRUMPPEPE.sol#84-86) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
TRUMPPEPE.constructor(address)._owner (TRUMPPEPE.sol#406) lacks a zero-check on :
        - taxWallet = _owner (TRUMPPEPE.sol#411)
TRUMPPEPE.changeTaxWallet(address)._wallet (TRUMPPEPE.sol#590) lacks a zero-check on :
        - taxWallet = _wallet (TRUMPPEPE.sol#591)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in TRUMPPEPE._transferFrom(address,address,uint256) (TRUMPPEPE.sol#482-500):
        External calls:
        - doContractSwap() (TRUMPPEPE.sol#489)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (TRUMPPEPE.sol#569-575)
                - (success) = taxWallet.call{value: swappedTokens}() (TRUMPPEPE.sol#554)
        External calls sending eth:
        - doContractSwap() (TRUMPPEPE.sol#489)
                - (success) = taxWallet.call{value: swappedTokens}() (TRUMPPEPE.sol#554)
        Event emitted after the call(s):
        - Transfer(sender,address(this),feeToken) (TRUMPPEPE.sol#517)
                - amountReceived = takeFee(sender,recipient,amount) (TRUMPPEPE.sol#495)
        - Transfer(sender,recipient,amountReceived) (TRUMPPEPE.sol#498)
Reentrancy in TRUMPPEPE.doContractSwap() (TRUMPPEPE.sol#546-560):
        External calls:
        - swapTokensForEth(contractTokenBalance) (TRUMPPEPE.sol#549)
                - router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (TRUMPPEPE.sol#569-575)
        - (success) = taxWallet.call{value: swappedTokens}() (TRUMPPEPE.sol#554)
        External calls sending eth:
        - (success) = taxWallet.call{value: swappedTokens}() (TRUMPPEPE.sol#554)
        Event emitted after the call(s):
        - ETHTransferFailed(taxWallet,swappedTokens) (TRUMPPEPE.sol#557)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Context._contextSuffixLength() (TRUMPPEPE.sol#31-33) is never used and should be removed
Context._msgData() (TRUMPPEPE.sol#27-29) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Pragma version^0.8.20 (TRUMPPEPE.sol#2) necessitates a version too recent to be trusted. Consider deploying with 0.8.18.
solc-0.8.20 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in TRUMPPEPE.doContractSwap() (TRUMPPEPE.sol#546-560):
        - (success) = taxWallet.call{value: swappedTokens}() (TRUMPPEPE.sol#554)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IUniswapV2Router01.WETH() (TRUMPPEPE.sol#214) is not in mixedCase
Parameter TRUMPPEPE.isFeeExcluded(address)._wallet (TRUMPPEPE.sol#542) is not in mixedCase
Parameter TRUMPPEPE.setDoContractSwap(bool)._enabled (TRUMPPEPE.sol#584) is not in mixedCase
Parameter TRUMPPEPE.changeTaxWallet(address)._wallet (TRUMPPEPE.sol#590) is not in mixedCase
Parameter TRUMPPEPE.setAuthorizedWallets(address,bool)._wallet (TRUMPPEPE.sol#598) is not in mixedCase
Parameter TRUMPPEPE.setAuthorizedWallets(address,bool)._status (TRUMPPEPE.sol#598) is not in mixedCase
Constant TRUMPPEPE._name (TRUMPPEPE.sol#365) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TRUMPPEPE._symbol (TRUMPPEPE.sol#366) is not in UPPER_CASE_WITH_UNDERSCORES
Constant TRUMPPEPE._decimals (TRUMPPEPE.sol#367) is not in UPPER_CASE_WITH_UNDERSCORES
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 (TRUMPPEPE.sol#219) is too similar
to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (TRUMPPEPE.sol#220)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
TRUMPPEPE._totalSupply (TRUMPPEPE.sol#369) should be constant
TRUMPPEPE.buyTotalFee (TRUMPPEPE.sol#379) should be constant
TRUMPPEPE.sellTotalFee (TRUMPPEPE.sol#381) should be constant
TRUMPPEPE.taxEnabled (TRUMPPEPE.sol#386) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
TRUMPPEPE.pair (TRUMPPEPE.sol#384) should be immutable
TRUMPPEPE.router (TRUMPPEPE.sol#383) should be immutable
TRUMPPEPE.swapThreshold (TRUMPPEPE.sol#388) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:TRUMPPEPE.sol analyzed (7 contracts with 93 detectors), 31 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0x4a23ef561a99ae3b18f1f1689257b2bfb57ec237e9a9ca3f81aa143860d9330d

2- Approve Max (passed):
https://testnet.bscscan.com/tx/0xcaee76d31cd2a3265a4b3c6f54b52923a4016fd9072ff979dbb9ea8197775445

3- Enable Trading (passed):
https://testnet.bscscan.com/tx/0x84355b233c3b12e50e47b443dcac73113fbc500467a83e8a76680691f5135837

4- Set Authorized Wallets (passed):
https://testnet.bscscan.com/tx/0x93fc946b162387fa1a66f6948047775bbad06d7be8ea8bb11896b0cf748a5850

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: Enable Trading
Status: Open

Overview:
The OpenTrading 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 {
  isTradeEnabled = true;
  }
```

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 – **Local Variable Shadowing**
Severity: Low
function: _approve and allowance
Status: Open

Overview:

```
function _approve(address owner, address spender, uint256
amount) internal virtual {
require(owner != address(0), "ERC20: approve from the zero
address");
require(spender != address(0), "ERC20: 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 Visibility**

Severity: Low

function: mapping

Status: Open

Overview:

It's simply saying that no visibility was specified, so it's going with the default. This has been related to security issues in contracts.

```
mapping(address => uint256) _balances;
mapping(address => mapping(address => uint256))
```

Suggestion:

You can easily silence the warning by adding the mapping public:

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 ^0.8.20;

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.

```
event AddAuthorizedWallet(address holder, bool status);
event AutoLiquify(uint256 amountBNB, uint256 amountBOG);
```

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   Ⓜ expelee

✈ Expelee   in 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.

expelee

Building the Futuristic **Blockchain Ecosystem**