expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

NinjaPepe

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	1
Medium	0
Low	0
Informational	0

Centralization Risks

Owner Privileges	Description
Can Owner Set Taxes >25% ?	Not Detected
Owner Can enable trading ?	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

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**
KYC Verification	**-**
Audit Date	**11 Jan 2024**

CONTRACT DETAILS

Token Address:
0x596A1017af27593e0AB371F74e36dC923f3B2C11

Name: NinjaPepe

Symbol: $NINJAPEPE

Decimals: 18

Network: BscScan

Token Type: BEP-20

Owner:
0x791577974dB7167289944922fB9C0D2D06Cd5EB4

Deployer:
0x79a6b0FFad7A415bCdf84FAA232ba8D54dbfF17f

Token Supply:
1,000,000,000,000

Checksum:
Be1c3a4fbb6e83e8393a57617b5a5b11

Testnet:
https://testnet.bscscan.com/address/0xbfb225a97341045cc1067ffee2effb0bd56b0c5a#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 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.

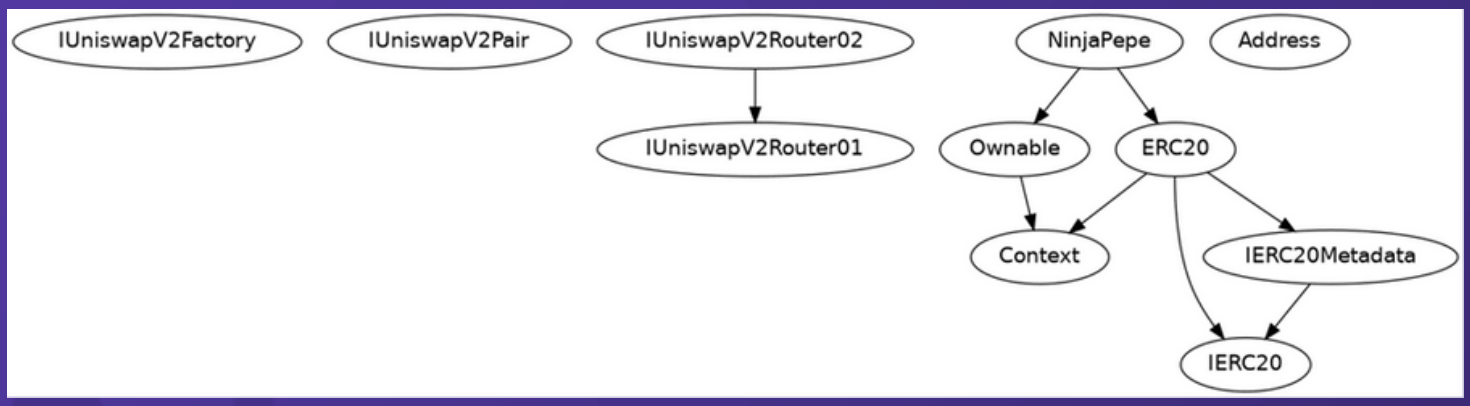
Low Risk

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

STATIC ANALYSIS

```
INFO:Detectors:
NinjaPepe.constructor().router (NinjaPepe.sol#548) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables
INFO:Detectors:
NinjaPepe.claimStuckTokens(address) (NinjaPepe.sol#593-602) ignores return value by address(msg.sender).sendValue(address(this).balance) (NinjaPepe.sol#596)
NinjaPepe.swapAndSendMarketing(uint256) (NinjaPepe.sol#690-709) ignores return value by address(marketingWallet).sendValue(newBalance) (NinjaPepe.sol#706)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Reentrancy in NinjaPepe._transfer(address,address,uint256) (NinjaPepe.sol#632-683):
        External calls:
        - swapAndSendMarketing(contractTokenBalance) (NinjaPepe.sol#657)
                - (success) = recipient.call{value: amount}() (NinjaPepe.sol#236)
                - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (NinjaPepe.sol#697-70
2)
                - address(marketingWallet).sendValue(newBalance) (NinjaPepe.sol#706)
        External calls sending eth:
        - swapAndSendMarketing(contractTokenBalance) (NinjaPepe.sol#657)
                - (success) = recipient.call{value: amount}() (NinjaPepe.sol#236)
        Event emitted after the call(s):
        - Transfer(sender,recipient,amount) (NinjaPepe.sol#479)
                - super._transfer(from,to,amount) (NinjaPepe.sol#682)
        - Transfer(sender,recipient,amount) (NinjaPepe.sol#479)
                - super._transfer(from,address(this),fees) (NinjaPepe.sol#679)
Reentrancy in NinjaPepe.swapAndSendMarketing(uint256) (NinjaPepe.sol#690-709):
        External calls:
        - uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(tokenAmount,0,path,address(this),block.timestamp) (NinjaPepe.sol#697-702)
        - address(marketingWallet).sendValue(newBalance) (NinjaPepe.sol#706)
        Event emitted after the call(s):
        - SwapAndSendMarketing(tokenAmount,newBalance) (NinjaPepe.sol#708)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Address._revert(bytes,string) (NinjaPepe.sol#327-339) uses assembly
        - INLINE ASM (NinjaPepe.sol#332-335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (NinjaPepe.sol#37) is not in mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (NinjaPepe.sol#38) is not in mixedCase
Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (NinjaPepe.sol#54) is not in mixedCase
Function IUniswapV2Router01.WETH() (NinjaPepe.sol#73) is not in mixedCase
Parameter NinjaPepe.changeMarketingWallet(address)._marketingWallet (NinjaPepe.sol#615) is not in mixedCase
Parameter NinjaPepe.setSwapEnabled(bool)._enabled (NinjaPepe.sol#685) 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 (NinjaPepe.sol#78) is too similar t
o IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (NinjaPepe.sol#79)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Detectors:
NinjaPepe.marketingFeeOnBuy (NinjaPepe.sol#532) should be immutable
NinjaPepe.marketingFeeOnSell (NinjaPepe.sol#533) should be immutable
NinjaPepe.swapTokensAtAmount (NinjaPepe.sol#537) should be immutable
NinjaPepe.uniswapV2Pair (NinjaPepe.sol#526) should be immutable
NinjaPepe.uniswapV2Router (NinjaPepe.sol#525) should be immutable
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-immutable
INFO:Slither:NinjaPepe.sol analyzed (11 contracts with 93 detectors), 38 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0x6e48b396c6accfb5cf35f92dc86fb52eb08400af52947c218876adfdf0913d6a

2- Increase Allowance (passed):
https://testnet.bscscan.com/tx/0x1226e24695fec0d786bc63e0f738de04a75a02adbe0ccf8d95d48c7eea6d806f

3- Decrease Allowance (passed):
https://testnet.bscscan.com/tx/0x5f2f0e8a61587a8cb4c95055f34be4aa42ebfc5bb5ab315c463ca4f60d340ecb

4- Enable Trading (passed):
https://testnet.bscscan.com/tx/0xd17a88581f753e32612d49224c55750e03ab10050075810c72fb3f2b23461a0e

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

Enabling Trades

Category: Centralization
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 already enabled.");
  tradingEnabled = true;
  swapEnabled = 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 provide investors with more confidence in the eventual activation of trading capabilities, mitigating concerns of potential bad faith actions by the original owner.

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.