expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

ElonX

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	0
Medium	0
Low	2
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

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	**8 august 2023**

CONTRACT DETAILS

Token Name: ElonX

Symbol: ElonX

Network: BSC

Language: Solidity

Contract Address: 0x1417EFe42Be82aC572e754aA85CeeaB4c0D69f63

Total Supply: 100,000,000

Owner's Wallet: 0x4ffDe3ca826fD5400445938A4d9046c65c6568b0

Deployer's Wallet: 0x4ffDe3ca826fD5400445938A4d9046c65c6568b0

Testnet.
https://testnet.bscscan.com/address/0xA8B082785571b7D6C36Ef1F06d687dc7BAD961e6

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.

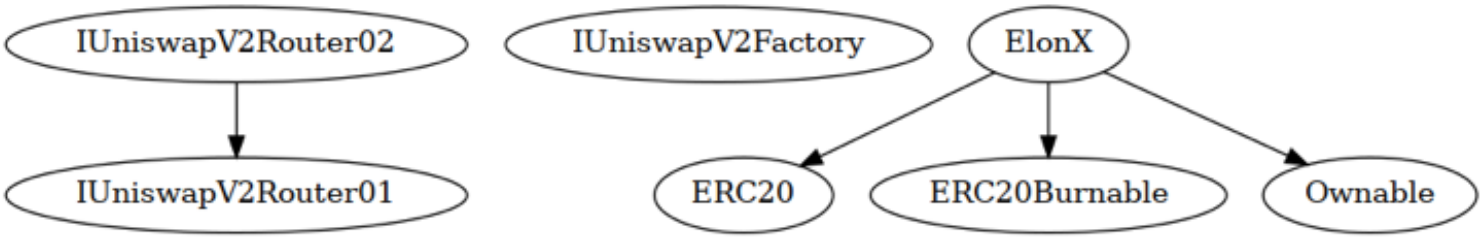
Low Risk

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

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IUniswapV2Router01	Interface			
└	factory	External ❗		NO ❗
└	WETH	External ❗		NO ❗
└	addLiquidity	External ❗	🛑	NO ❗
└	addLiquidityETH	External ❗	💵	NO ❗
└	removeLiquidity	External ❗	🛑	NO ❗
└	removeLiquidityETH	External ❗	🛑	NO ❗
└	removeLiquidityWithPermit	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO ❗
└	swapExactTokensForTokens	External ❗	🛑	NO ❗
└	swapTokensForExactTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokens	External ❗	💵	NO ❗
└	swapTokensForExactETH	External ❗	🛑	NO ❗
└	swapExactTokensForETH	External ❗	🛑	NO ❗
└	swapETHForExactTokens	External ❗	💵	NO ❗
└	quote	External ❗		NO ❗
└	getAmountOut	External ❗		NO ❗
└	getAmountIn	External ❗		NO ❗
└	getAmountsOut	External ❗		NO ❗
└	getAmountsIn	External ❗		NO ❗
IUniswapV2Router02	Interface	IUniswapV2Router01		
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO ❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO ❗
IUniswapV2Factory	Interface			
└	feeTo	External ❗		NO ❗
└	feeToSetter	External ❗		NO ❗
└	getPair	External ❗		NO ❗
└	allPairs	External ❗		NO ❗

FUNCTION DETAILS

└	allPairsLength	External ❗		NO ❗
└	createPair	External ❗	🛑	NO ❗
└	setFeeTo	External ❗	🛑	NO ❗
└	setFeeToSetter	External ❗	🛑	NO ❗

ElonX	Implementation	ERC20, ERC20Burnable, Ownable		
└	<Constructor>	Public ❗	🛑	ERC20
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setTransferTax	External ❗	🛑	onlyOwner
└	isExcludedFromFee	External ❗		NO ❗
└	excludeFromFee	External ❗	🛑	onlyOwner
└	includeInFee	External ❗	🛑	onlyOwner
└	collectMarketingFees	Public ❗	🛑	NO ❗
└	swapAndLiquidity	Public ❗	🛑	lockTheSwap
└	percentOf	Public ❗		NO ❗
└	_transfer	Internal 🔒	🛑	
└	recover	Public ❗	🛑	onlyOwner
└	initStaking	External ❗	🛑	onlyOwner
└	initAirdrop	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💵	NO ❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

TESTNET VERSION

Adding Liquidity ✅
Tx:
https://testnet.bscscan.com/tx/0xe62150c310eb4f2a00521734ec94cdc4cc12ec990e5341ac9cac7c00e968fa46

==========

Buying when excluded from fees ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x3e21ce9167ed7d7899e8575e1b0ee209f7a2ba3abe464abe988eed37753e865b

==========

Selling when excluded from fees ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0xef6d6242754b0229f3e949d082f73540cf905af2f3a3c1bb79b38a5332cf2542

==========

Transferring when excluded from fees ✅
Tx (0% tax):
https://testnet.bscscan.com/tx/0x31dd6c2a8d1c90ad108067d453a5c86529683ef8e65f84d878e0676ed8544d15

==========

Buying ✅
Tx (0-0.1% tax):
https://testnet.bscscan.com/tx/0xc3a0d499870f207fb9dcea05ff7bc480bea8b067e20a233868c733712cb201cf

TESTNET VERSION

Selling
Tx (0-0.1% tax):
https://testnet.bscscan.com/tx/0x9803fc525911ab412996478130a4088e166c98f6e4b2c3047607bc39ef378bf2

==========

Transferring
Tx (0% tax):
https://testnet.bscscan.com/tx/0x9f35f6357eabe661e80c070d01199f788f3eeeecf362263cf1b2ee4580917aa0

==========

Internal swap (BNB to marketing wallet | reward token to dividend tracker | reward distribution)
Tx:
https://testnet.bscscan.com/tx/0xe16b7e0a93a60e04a88cfceea42fd68369f1a39dba43fbb1b9e5dad4edee6737

withdrawing marketing ETH funds:
https://testnet.bscscan.com/tx/0x053506c254d74391c5a0cfaac7ffa3122116944f4cca086acefc8420cfba1da6

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			

LOW RISK FINDING

Minting to staking and airdrop contracts

Category: Centralization
Status: Open
Impact: Unknown (Low - Critical)

Overview:
The initAirdrop and initStaking functions utilize the create2 opcode to instantiate airdrop and staking contracts at predetermined addresses. Subsequently, these functions mint a significant portion of the total token supply to these addresses. Specifically, upon deployment, the staking contract receives 25% of the total supply, and the airdrop contract obtains 15% of the total supply. Allocating such a significant fraction of the total token supply to these contracts could potentially lead to issues which are currently outside the purview of this audit.

Suggestion:
Given the substantial proportion of the total supply allocated to these contracts, it is highly recommended to conduct a thorough audit of both the staking and airdrop contracts. This will help in identifying potential risks, ensuring the security of these contracts, and safeguarding against potential centralization issues. Careful review and perhaps, modifications in the token allocation percentages might also be considered to mitigate centralization risks.

LOW RISK FINDING

Predictable Salt in Contract Creation Leading to Potential Exploits

Category: Front-Running Vulnerability
Status: Open
Impact: Unknown (Low - Critical)

Overview:
The initAirdrop and initStaking functions in the code use the create2 opcode to construct airdrop and staking contracts at predetermined addresses, based on provided input. The usage of a predictable salt opens up possibilities for exploitation through front-running. A malicious actor monitoring the transaction pool could potentially identify pending transactions related to these function calls and could pursue one of the following strategies to exploit the system:

1. Deploy an airdrop or staking contract at the anticipated address. This would result in a reversion of initAirdrop or initStaking function calls.
2. Exploit the initial state of the newly deployed staking or airdrop contracts within the same block and timestamp as their deployment. The specific attack vectors would depend on the implementation details of the staking and airdrop contracts, which are outside the scope of this audit. For instance, an attacker might leverage a flash loan to launch an attack on the staking contract.

Suggestion:
Suggestion:
It is recommended to incorporate robust safety measures within the airdrop and staking contracts. Care should be taken to ensure that the initial state of these contracts does not expose any significant vulnerabilities or opportunities for exploitation. This might include mechanisms to prevent unexpected or unauthorized contract creation and additional controls on state changes following contract deployment.

INFORMATINAL RISK FINDING

Updating buy/sell fees

Category: Coding mistake
Status: Open
Impact: Informational
Overview:
iThe "setTransferTax" function is intended to update the fees for buying and selling within the smart contract. The function requires the input fees for buying and selling to be less than or equal to 10 whereas the denominator which is used for calculating fees is 10_000, this means, 10 is actually 0.1% not 10%

```
function setTransferTax(
  uint256 _buyFee,
  uint256 _sellFee
) external onlyOwner {
  require(_buyFee <= 10, "Buy Fee cannot be more than 10%");
  require(_sellFee <= 10, "Sell Fee cannot be more than 10%");
  buyFee = _buyFee;
  sellFee = _sellFee;
}
```

INFORMATINAL RISK FINDING

Suggestion:
update require statement to match error message, or correct error message to match require statement

```
function setTransferTax(
  uint256 _buyFee,
  uint256 _sellFee
) external onlyOwner {
  require(_buyFee <= 10, "Buy Fee cannot be more than 0.1%");
  require(_sellFee <= 10, "Sell Fee cannot be more than 0.1%");
  buyFee = _buyFee;
  sellFee = _sellFee;
}
```

```
function setTransferTax(
  uint256 _buyFee,
  uint256 _sellFee
) external onlyOwner {
  require(_buyFee <= 100, "Buy Fee cannot be more than 1%");
  require(_sellFee <= 100, "Sell Fee cannot be more than 1%");
  buyFee = _buyFee;
  sellFee = _sellFee;
}
```

INFORMATINAL RISK FINDING

swapAndLiquidity

Category: Documentation
Status: Open
Impact: Informational
Overview:
The comments written in the contract, indicates that swapAndLiquidity function should work in an automated manner. But this function is public and is not in _trasnfer function

```
 //Swap Tokens for ETH or to add liquidity either automatically or manual, due to SAFU this was changed to Automatic after enable trading.
 //Corrected newBalance bug, it sending bnb to wallet and any remaining is on contract and can be recoverred.
 function swapAndLiquidity() public lockTheSwap {
  uint256 contractBalance = balanceOf(address(this));

  if (contractBalance > 0) {
   // generate the uniswap pair path of token -> weth
   address[] memory path = new address[](2);
   path[0] = address(this);
   path[1] = uniswapV2Router.WETH();

   // make the swap
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
      contractBalance,
      0, // accept any amount of ETH
      path,
      address(this), // The contract
      block.timestamp
    );
  }
 }
```

INFORMATINAL RISK FINDING

Suggestion:
If this implementation is not intentional, make sure to change swapAndLiquidity function to internal and include it in _transfer function

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.