expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

SORA

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	0
Medium	0
Low	3
Informational	1

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**
Audit Date	**15 March 2024**

CONTRACT DETAILS

Token Address: 0x47531F97454a1Cac53447fa58233be401fE67f19

Name: SORA

Symbol: SORA

Decimals: 9

Network: BSC

Token Type:BEP-20

Owner: 0x8527b3CfB18264C881D3639c23b7a55371424fa7

Deployer: 0x8527b3CfB18264C881D3639c23b7a55371424fa7

Token Supply: 21000000

Checksum: A2032c616934aeb47e6039f76b20d213

Testnet:
https://testnet.bscscan.com/address/0x1286b30bcbfede43b2d51fbd145a8b6deae4d90a#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

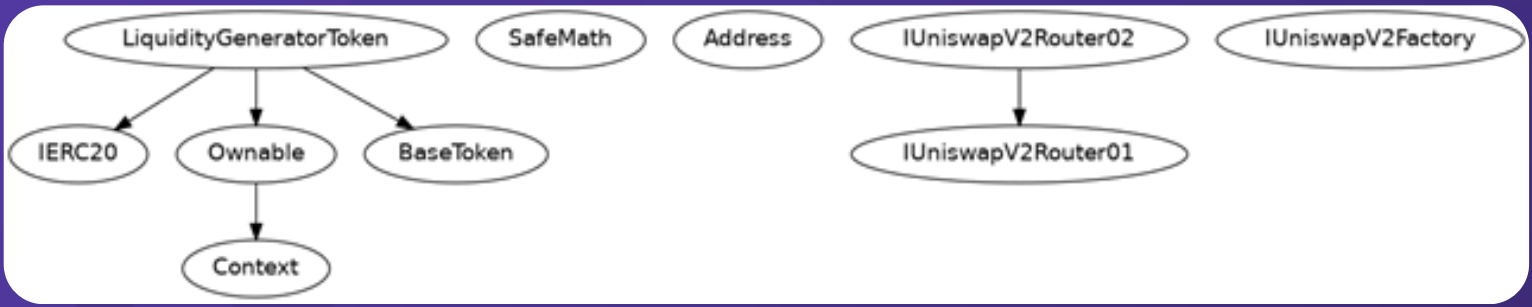
Issues on this level are minor details and warning that can remain unfixed.

Informational

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

expelee

INHERITANCE TREES



expelee

STATIC ANALYSIS

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

```
INFO:Detectors:
LiquidityGeneratorToken.addLiquidity(uint256,uint256) (LiquidityGeneratorToken.sol#1541-1554) ignores return value by uniswapV2Router.addLiquidityETH{value:
 ethAmount}(address(this),tokenAmount,0,0,address(0xdead),block.timestamp) (LiquidityGeneratorToken.sol#1546-1553)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
LiquidityGeneratorToken.allowance(address,address).owner (LiquidityGeneratorToken.sol#1078) shadows:
        - Ownable.owner() (LiquidityGeneratorToken.sol#150-152) (function)
LiquidityGeneratorToken._approve(address,address,uint256).owner (LiquidityGeneratorToken.sol#1449) shadows:
        - Ownable.owner() (LiquidityGeneratorToken.sol#150-152) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
LiquidityGeneratorToken.setTaxFeePercent(uint256) (LiquidityGeneratorToken.sol#1241-1247) should emit an event for:
        - _taxFee = taxFeeBps (LiquidityGeneratorToken.sol#1242)
LiquidityGeneratorToken.setLiquidityFeePercent(uint256) (LiquidityGeneratorToken.sol#1249-1258) should emit an event for:
        - _liquidityFee = liquidityFeeBps (LiquidityGeneratorToken.sol#1253)
LiquidityGeneratorToken.setCharityFeePercent(uint256) (LiquidityGeneratorToken.sol#1260-1266) should emit an event for:
        - _charityFee = charityFeeBps (LiquidityGeneratorToken.sol#1261)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-arithmetic
INFO:Detectors:
LiquidityGeneratorToken.constructor(string,string,uint256,address,address,uint16,uint16,uint16,address,uint256).serviceFeeReceiver_ (LiquidityGeneratorToken
.sol#987) lacks a zero-check on :
                - address(serviceFeeReceiver_).transfer(serviceFee_) (LiquidityGeneratorToken.sol#1045)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
```

```
INFO:Detectors:
Address.isContract(address) (LiquidityGeneratorToken.sol#445-455) uses assembly
        - INLINE ASM (LiquidityGeneratorToken.sol#451-453)
Address.verifyCallResult(bool,bytes,string) (LiquidityGeneratorToken.sol#614-634) uses assembly
        - INLINE ASM (LiquidityGeneratorToken.sol#626-629)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
LiquidityGeneratorToken.includeInReward(address) (LiquidityGeneratorToken.sol#1200-1211) has costly operations inside a loop:
        - _excluded.pop() (LiquidityGeneratorToken.sol#1207)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#costly-operations-inside-a-loop
INFO:Detectors:
Address.functionCall(address,bytes) (LiquidityGeneratorToken.sol#498-500) is never used and should be removed
Address.functionCall(address,bytes,string) (LiquidityGeneratorToken.sol#508-514) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (LiquidityGeneratorToken.sol#527-533) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (LiquidityGeneratorToken.sol#541-552) is never used and should be removed
Address.functionDelegateCall(address,bytes) (LiquidityGeneratorToken.sol#587-589) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (LiquidityGeneratorToken.sol#597-606) is never used and should be removed
Address.functionStaticCall(address,bytes) (LiquidityGeneratorToken.sol#560-562) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (LiquidityGeneratorToken.sol#570-579) is never used and should be removed
Address.isContract(address) (LiquidityGeneratorToken.sol#445-455) is never used and should be removed
Address.sendValue(address,uint256) (LiquidityGeneratorToken.sol#473-478) is never used and should be removed
Address.verifyCallResult(bool,bytes,string) (LiquidityGeneratorToken.sol#614-634) is never used and should be removed
Context._msgData() (LiquidityGeneratorToken.sol#110-112) is never used and should be removed
SafeMath.div(uint256,uint256,string) (LiquidityGeneratorToken.sol#380-389) is never used and should be removed
SafeMath.mod(uint256,uint256) (LiquidityGeneratorToken.sol#340-342) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (LiquidityGeneratorToken.sol#406-415) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (LiquidityGeneratorToken.sol#211-217) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (LiquidityGeneratorToken.sol#253-258) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (LiquidityGeneratorToken.sol#265-270) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (LiquidityGeneratorToken.sol#236-246) is never used and should be removed
SafeMath.trySub(uint256,uint256) (LiquidityGeneratorToken.sol#224-229) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version=0.8.17 (LiquidityGeneratorToken.sol#911) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
```

```
INFO:Detectors:
Pragma version>=0.6.0<0.9.0 (Token.sol#6) is too complex
solc-0.8.22 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Bonki.contractSwap(uint256) (Token.sol#511-567):
        - (success,None) = _taxWallets.marketing.call{gas: 55000,value: marketingBalance}() (Token.sol#562)
        - (success,None) = _taxWallets.buyback.call{gas: 55000,value: buybackBalance}() (Token.sol#565)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Function IRouter01.WETH() (Token.sol#37) is not in mixedCase
Parameter Bonki.setProtectionSettings(bool,bool)._antiSnipe (Token.sol#376) is not in mixedCase
Parameter Bonki.setProtectionSettings(bool,bool)._antiBlock (Token.sol#376) is not in mixedCase
Constant Bonki.startingSupply (Token.sol#116) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Bonki._name (Token.sol#117) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Bonki._symbol (Token.sol#118) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Bonki._decimals (Token.sol#119) is not in UPPER_CASE_WITH_UNDERSCORES
Constant Bonki._tTotal (Token.sol#120) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Bonki._taxRates (Token.sol#135-139) is not in mixedCase
Variable Bonki._ratios (Token.sol#141-146) is not in mixedCase
Constant Bonki.masterTaxDivisor (Token.sol#151) is not in UPPER_CASE_WITH_UNDERSCORES
Variable Bonki._taxWallets (Token.sol#163-166) is not in mixedCase
Variable Bonki._hasLiqBeenAdded (Token.sol#175) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Variable IRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (Token.sol#49) is too similar to IRouter01.addLiquidity(address,address,uint256,uint256,uint256,u
int256,address,uint256).amountBDesired (Token.sol#50)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-too-similar
INFO:Slither:Token.sol analyzed (7 contracts with 93 detectors), 47 result(s) found
```

TESTNET VERSION

1- Approve (passed):
https://testnet.bscscan.com/tx/0x6ed22c83f307b1cbac66407881cdc18c579f76b794e252ebdc5c9725c8c3f89c

2- Set Charity Fee Percent (passed):
https://testnet.bscscan.com/tx/0xb49781876fe5fbce17c51c78635ef64965157acd9b19f5b64fa1db41090a3db1

3- Set Liquidity Fee Percent (passed):
https://testnet.bscscan.com/tx/0x4b0f214e6adc79759ae9b2252374700a08e9b23b0ad04790912f2adf09585620

4- Set Tax Fee Percent (passed):
https://testnet.bscscan.com/tx/0xa5120edd7f3c7ce3c593e08eeacf5be32495774e6d8d4c02ece40d08ca6d30e6

5- Set Tax Fee Percent (passed):
https://testnet.bscscan.com/tx/0xdfeb050766c3575472e2b54bb7dd8070b0e6594313fdcd26c3396f95d09bb434

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

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 setTaxFeePercent(uint256 taxFeeBps) external onlyOwner {
  _taxFee = taxFeeBps;
require(
   _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
"Total fee is over 25%"
  );
 }
function setLiquidityFeePercent(uint256 liquidityFeeBps)
external
  onlyOwner
 {
  _liquidityFee = liquidityFeeBps;
require(
   _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
"Total fee is over 25%"
  );
 }
 function setCharityFeePercent(uint256 charityFeeBps) external
onlyOwner {
  _charityFee = charityFeeBps;
require(
   _taxFee + _liquidityFee + _charityFee <= MAX_FEE,
"Total fee is over 25%"
  );
 }
```

LOW RISK FINDING

Centralization – Missing Visibility

Severity: Low

function: Visibility

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.

```
bool inSwapAndLiquify;
```

Suggestion

You can easily silence the warning by adding the public/private

LOW RISK FINDING

Centralization – Local variable Shadowing

Severity: Low

function: Variable Shadowing

Status: Open

Overview:

```
  function allowance(address owner, address spender)
public
view
  override
returns (uint256)
 {
return _allowances[owner][spender];
 }
function _approve(
address owner,
address spender,
uint256 amount
 ) private {
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 variables that shadow another component.

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 MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
```

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.