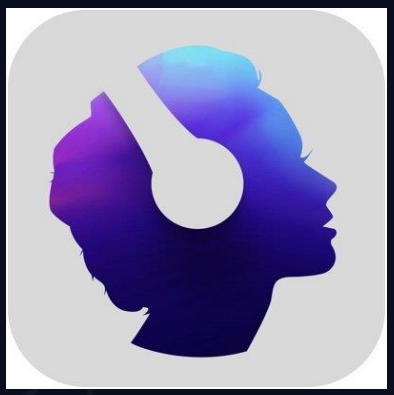
expelee

A Secure Place For Web3

SMART CONTRACT AUDIT OF

Soundtopia Fair Launch



Contract Address

0xd03de35af6D507708947929A36eB6841872aff79

Audit Summary

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**

Ownership: NOT **RENOUNCED**

KYC Verification: Done

Audit Date: 20/07/2022

Audit Team: **EXPELEE**

Be aware that smart contracts deployed on the blockchain aren't resistant to internal exploit, external vulnerability, or hack. For a detailed understanding of risk severity, source code vulnerability, functional hack, and audit disclaimer, kindly refer to the audit.

DISCLAMER

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 guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document.

Always Do your own research and protect 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. Always 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.

Contract Review

Contract Name	**Sount**
Compiler Version	**v0.8.11+commit.d7f03943**
Optimization	**Yes with 200 runs**
License	**Unlicense**
Explorer	**https://bscscan.com/address/0xd03de35af6D507708947929A36eB6841872aff79#code**
Symbol	**SOUNT**
Decimals	**18**
Total Supply	**1,000,000,000**
Domain	**https://soundtopia.app/**

Project Review

Token Name: Soundtopia

Web Site: https://soundtopia.app/

Twitter: https://twitter.com/Soundtopia_

Telegram: https://t.me/Soundtopia_Official

Contract Address:
0xd03de35af6D507708947929A36eB6841872aff79

Platform: Binance Smart Chain

Token Type: BEP 20

Language: SOLIDITY

Audit Methodology

The scope of this report is to audit the smart contract source code. We have scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Category

Smart Contract Vulnerabilities	- Unhandled Exceptions - Transaction Order Dependency - Integer Overflow - Unrestricted Action - Incorrect Inheritance Order - Typographical Errors - Requirement Violation
Source Code Review	- Gas Limit and Loops - Deployment Consistency - Repository Consistency - Data Consistency - Token Supply Manipulation
Functional Assessment	- Operations Trail & Event Generation - Assets Manipulation - Liquidity Access

Vulnerability Checklist

№	Description.	Result
1	Compiler warnings.	Passed
2	Race conditions and Re-entrancy. Cross-function raceconditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

Low-Risk

4 low-risk code issues found

Medium-Risk

0 medium-risk code issues found

High-Risk

0 high-risk code issues found

Audit Summary

```
Number of lines: 1398 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 10 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 34
Number of informational issues: 38
Number of low issues: 4
Number of medium issues: 0
Number of high issues: 0
ERCs: ERC2612, ERC20
```

Name	# functions	ERCS	ERC20 info	Complex code	Features
SafeMath	8			No	
Address	7			No	Send ETH Assembly
IUniswapV2Factory	8			No	
IUniswapV2Pair	27	ERC20,ERC2612	∞ Minting Approve Race Cond.	No	
IUniswapV2Router02	24			No	Receive ETH
SOUNT	86	ERC20	No Minting Approve Race Cond.	No	Receive ETH Send ETH

Low-Risk

1) Functions that send Ether to arbitrary destinations

Unprotected call to a function sending Ether to an arbitrary address.

```
function sendMarketingDividends(uint256 amount) private {
        (bool success,) = payable(address(marketingWallet)).call{value: amount}("");

        if(success) {
            emit SendMarketingDividends(amount);
        }
    }}
```

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

2) Reentrancy vulnerabilities

Detection of the reentrancy bug. Do not report reentrancies that don't involve Ether (see reentrancy-no-eth)

```
    uniswapV2Router.addLiquidityETH{value: bnbAmount}(

            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            liquidityWallet, // send to liquidity wallet
            block.timestamp
    );
```

Recommendation

Rename Apply the check-effects-interactions pattern.

3) Unused return

The return value of an external call is not stored in a local or state variable.

```
function _addLiquidity(uint256 tokenAmount, uint256 bnbAmount) private {
        // approve token transfer to cover all possible scenarios
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        // add the liquidity
        uniswapV2Router.addLiquidityETH{value: bnbAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            liquidityWallet, // send to liquidity wallet
            block.timestamp
        );
    }
```

Recommendation

Ensure that all the return values of the function calls are used.

4) Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
        function setTradingEnabledTimestamp(uint256 timestamp) external onlyOwner {
        require(tradingEnabledTimestamp > block.timestamp, "Changing the timestamp is not allowed
if the listing   has already started");
        tradingEnabledTimestamp = timestamp;
    }
```

Recommendation

Emit an event for critical parameter changes.

Manual Audit (Contract Function)

```
contract SOUNT is Context, IERC20, Ownable {
    using SafeMath for uint256;
    using Address for address;

    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private _allowances;

    mapping (address => bool) private _isExcludedFromFee;

    mapping (address => bool) private _isExcludedFromReward;

    mapping (address => bool) private _isExcludedFromMaxSellTransactionAmount;
    address[] private _excludedFromReward;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal = 1000000000* 10**18;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));

    address public liquidityWallet;
    address payable public marketingWallet = payable(0xffF03b028a79620Ee40AB3E120D22DF8026f4d03);

    uint256 private _tFeeTotal;

    string private _name = "Soundtopia";
    string private _symbol = "SOUNT";
    uint8 private _decimals = 18;

    address constant private  DEAD = 0x000000000000000000000000000000000000dEaD;

    uint8 public sellRewardFee = 4;
    uint8 public buyRewardFee = 2;

    uint8 public sellLiquidityFee = 4;
    uint8 public buyLiquidityFee = 2;

    uint8 public sellMarketingFee = 4;
    uint8 public buyMarketingFee = 4;

    uint8 totalSellFees;
    uint8 totalBuyFees;

    IUniswapV2Router02 public uniswapV2Router;
    address public uniswapV2Pair;

    bool private _inSwapAndLiquify;
    bool public swapAndLiquifyEnabled = true;
```

```
    uint256 public maxSellTransactionAmount = 5000000 *10**18;
    uint256 private _swapTokensAtAmount =  1000000* 10**18;

    uint256 public tradingEnabledTimestamp = 1658307600;
    // addresses that can make transfers before PancakeSwap listing
    mapping (address => bool) private _canTransferBeforeTradingIsEnabled;
    // exclude from transactions
    mapping(address=>bool) private _isBlacklisted;
    // all known liquidity pools
    mapping (address => bool) public automatedMarketMakerPairs;

    uint256 private _marketingCurrentAccumulatedFee;
    uint256 private _liquidityCurrentAccumulatedFee;

    event MinTokensBeforeSwapUpdated(uint256 minTokensBeforeSwap);
    event SwapAndLiquifyEnabledUpdated(bool enabled);
    event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
```

Important Points To Consider

✓ Verified contract source

✓ Token is sellable (not a honeypot) at this time

✗ Ownership not renounced or source contain an ownership functionality

✗ Source does contain a fee modifier

Source contain blocklist capability

✓ Buy fee is less than 10% (9%)

✓ Sell fee is less than 10% (8.9%)

✓ Owner/creator wallet contains less than 10% of circulating token supply (3.75%)

About Expelee

Expelee is a community driven organisation dedicated to fostering an anti-rug movement. We're here to keep investment safe from fraudsters. We've encountered several rug pulls and know how it feels to be duped, which is why we don't want anybody else to go through the same experience. We are here to raise awareness through our services so that the future of cryptocurrency can be rug-free.

The auditing process focuses to the following considerations with collaboration of an expert team:

- Functionality test of the Smart Contract to determine if proper logic has been followed throughout the whole process.
- Manually detailed examination of the code line by line by experts.
- Live test by multiple clients using Test net.
- Analysing failure preparations to check how the Smart
- Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analysing the security of the on-chain data.

Social Media

/Expelee

/ExpeleeOfficial

/expelee-co