expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

SonicSwapFactory

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	0
Medium	1
Low	3
Informational	0

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	**13 April 2025**

CONTRACT DETAILS

Token Address: -

Name: SonicSwapFactory

Symbol: -

Decimals:-

Network: -

Token Type:-

Owner: -

Deployer: -

Token Supply: -

Checksum: -

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

- Manual Review: The code has undergone a line-by-line review by the Ace team.
- BSC Test Network: All tests were conducted on the BSC Test network, and each test has a corresponding transaction attached to it. These tests can be found in the "Functional Tests" section of the report.
- Slither: The code has undergone static analysis using Slither.

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.

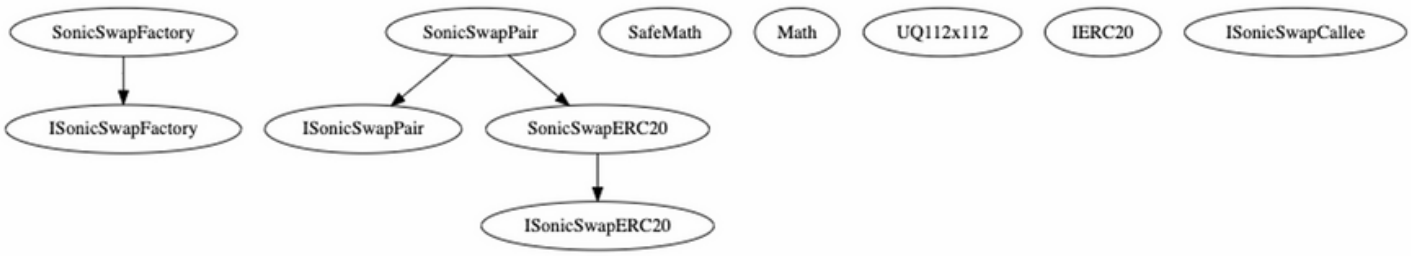
Low Risk

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 TREE



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			

MEDIUM RISK FINDING

Centralization – Centralized Control of Fee Parameters

Severity: Medium

Overview:

The protocol fee mechanism is controlled by a single address (feeToSetter) with no time-delay or multi-signature requirements. If this address is compromised, an attacker could immediately redirect protocol fees to an address they control, potentially extracting significant value from the protocol.

```
function setFeeTo(address _feeTo) external { require(msg.sender == feeToSetter, "SonicxSwap: FORBIDDEN"); feeTo = _feeTo; } function setFeeToSetter(address _feeToSetter) external { require(msg.sender == feeToSetter, "SonicxSwap: FORBIDDEN"); feeToSetter = _feeToSetter; }
```

Recommendation: Consider implementing a timelock mechanism for fee changes and requiring multi-signature authorization to reduce centralization risk and improve governance security.

LOW RISK FINDING

Centralization – Missing Events for Critical Parameter Changes

Severity: Low

Suggestion:

Emit events when changing critical parameters, making it impossible to track changes off-chain.

```
function setFeeTo(address _feeTo) external {
    require(msg.sender == feeToSetter, "SonicxSwap: FORBIDDEN");
    feeTo = _feeTo;
}
function setFeeToSetter(address _feeToSetter) external {
    require(msg.sender == feeToSetter, "SonicxSwap: FORBIDDEN");
    feeToSetter = _feeTo
```

Suggestion: Add event emissions for all protocol parameter changes to improve transparency and enable proper tracking of administrative actions

LOW RISK FINDING

Centralization – Missing Zero-Address Validation

Severity: Low

Description: There's no check preventing the feeToSetter address from being set to address(0). Setting feeToSetter to address(0) would permanently disable the ability to change fee parameters, as no address would have the authority to call

setFeeTo or setFeeToSetter again. function setFeeToSetter(address _feeToSetter) external { require(msg.sender == feeToSetter, "SonicxSwap: FORBIDDEN"); feeToSetter = _feeToSetter; }

Recommendation: Add a require statement to prevent setting feeToSetter to the zero address, protecting against accidental protocol parameter lockout.

LOW RISK FINDING

pragma solidity = 0.5.16;
Severity: Low
Status: Open

Description:

The contract uses Solidity 0.5.16, which lacks important safety features available in newer versions. Solidity 0.8.x provides built-in overflow/underflow protection and custom error types that could improve contract security and gas efficiency without relying on external libraries.

pragma solidity =0.5.16;

Recommendation: Update to a more recent Solidity version (0.8.x) to benefit from built-in safety features and optimizations, reducing reliance on SafeMath and improving overall code quality

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.