expelee

Building the Futuristic Blockchain Ecosystem

SECURITY AUDIT REPORT

Sonic StakingFarmsFactory

TOKEN OVERVIEW

Risk Findings

Severity	Found
High	0
Medium	0
Low	4
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	**Low Risk Detected**
Audit Date	**13 April 2025**

CONTRACT DETAILS

Contract Address:
0x7D36Eb844cCD5682F7D4CB37168E4187A4D8c466

Contract Name: SonicxSwapStakingRewardsFactory

Blockchain: Sonic

Contract Type: ERC-20

Contract Creator:
0xa4c576e2373282e94ae08ee4212f552d9555b986

Compiler Version: v0.5.17+commit.d19bba13

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.

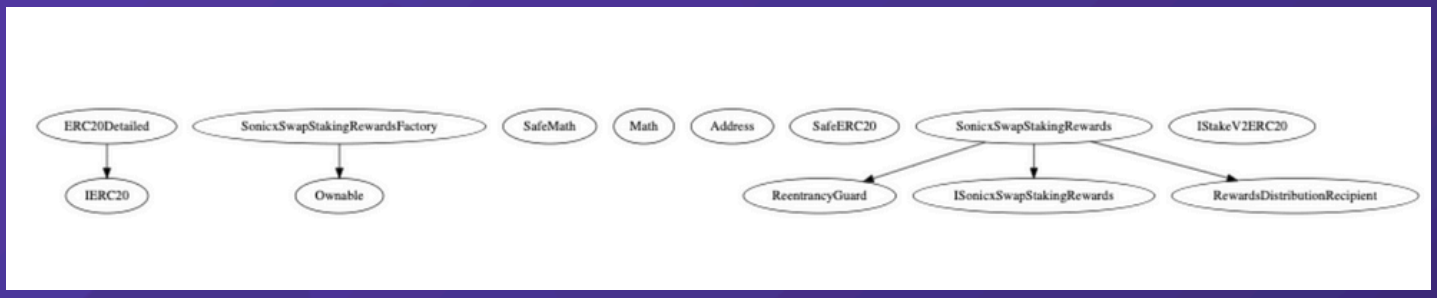
Low Risk

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



POINTS TO NOTE

- The owner can deploy new staking reward contracts
- The owner can renounce ownership
- The owner (of SonicxSwapStakingRewards) can withdraw reward tokens

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 – Inconsistent Token Transfer Method in adminWithdraw

Severity: Low

Suggestion:

The adminWithdraw function uses a direct token transfer method instead of the SafeERC20 wrapper used elsewhere in the contract. This inconsistency could lead to silent failures with non-standard tokens.

IERC20(rewardsToken).transfer(_toUser, _amount);

Recommendation: Replace with rewardsToken.safeTransfer(_toUser, _amount) to maintain consistency and handle all token types safely.

LOW RISK FINDING

Centralization – Potential for Owner to Undermine Reward Commitments

Severity: Low

Description: While the adminWithdraw function appears to be designed as a failsafe for emergency token recovery, it allows the owner to withdraw any amount of reward tokens without considering whether those tokens are needed to fulfill the current reward rate commitments. If the owner withdraws tokens after notifyRewardAmount has committed to a certain reward rate, the contract may have insufficient funds to pay all promised rewards.

```
function adminWithdraw(address _toUser, uint256 _amount) external returns (bool) { require(msg.sender == owner, "Only owner"); require(_toUser != address(0), "Invalid Address");
require(IERC20(rewardsToken).balanceOf(address(this)) >= _amount, "StakingFactory: insufficient amount"); IERC20(rewardsToken).transfer(_toUser, _amount); return true; }
```

Recommendation: Modify the adminWithdraw function to only allow withdrawal of "excess" tokens not needed for the current reward rate commitment. This could be implemented by calculating the tokens needed to fulfill the current reward rate until periodFinish and only allowing withdrawal of amounts above that threshold.

LOW RISK FINDING

Centralization – Missing Zero-Address Validation

Severity: Low

Description: The deploy function don't validate that critical addresses are not the zero address. If a zero address is provided for tokens, distribution, or owner parameters, it could lead to dysfunctional contracts or lost tokens.

function deploy(address stakingToken, address rewardToken, uint256 rewardAmount, uint256 _duration) public onlyOwner { StakingRewardsInfo storage info = stakingRewardsInfoByStakingToken[stakingToken]; require(info.stakingRewards == address(0), "StakingRewardsFactory::deploy: already deployed"); info.stakingRewards = address(new SonicxSwapStakingRewards(/*_rewardsDistribution=*/ address(this), rewardToken, stakingToken, _duration, msg.sender)); info.rewardAmount = rewardAmount; info.rewardToken = rewardToken; stakingTokens.push(stakingToken); emit _depoly(info.stakingRewards, rewardToken, stakingToken, rewardAmount); }

Recommendation: Add explicit zero address checks for all address parameters in the deploy function.

LOW RISK FINDING

Centralization – Outdated Solidity Version

Severity: Low

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 0.5.17;
```

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

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.