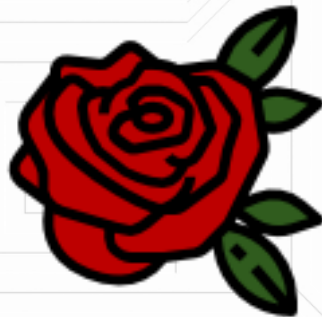# SMART CONTRACT SECURITY AUDIT OF

ROSE SWAP

SMART CONTRACT AUDIT | SOLIDITY DEVELOPMENT & TESTING | KYC | PROJECT EVALUATION

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN

Summary

Auditing Firm	InterFi Network
Client Firm	Rose Swap
Architecture	InterFi "Echelon" Auditing Standard
Language	Solidity
Mandatory Audit Check	Static, Software, Auto Intelligent & Manual Analysis
Final Report Date	January 15, 2022

Audit Summary

InterFi team has performed a line-by-line manual analysis and automated review of the smart contracts. Smart contracts were analyzed mainly for common vulnerabilities, exploits, and manipulation hacks. According to the contract audit:

❖ Rose Swap's smart contract source codes have **LOW RISK SEVERITY**

❖ Rose Swap has **PASSED** the smart contract audit

❖ Rose Swap uses **MINT** to generate governance tokens. Mint can be called by Minter (MasterChef.sol) and Owner (Timelock.sol)

For the detailed understanding of risk severity, and vulnerability, kindly refer to the audit. Please note, contracts make external calls and import various code packages to work effectively, InterFi does not provide explicit guarantee on the safety and security of these external calls/package imports.

📋 **RoseSwap.io DEX** is a fork of PancakeSwap.finance DEX but on Oasis Network

⚙ Blockchain: **Oasis Network**

✅ Verify the authenticity of this report on InterFi's GitHub: **https://github.com/interfinetwork**

Table Of Contents

Project Overview

InterFi was consulted by Rose Swap to conduct the smart contract security audit of their solidity source codes.

About Rose Swap

Rose Swap is an AMM, Yield Farming pool, and community governed DAO hosted on Oasis Network.

Project	Rose Swap
Blockchain	Oasis Network
Language	Solidity
Contract	https://github.com/roseswapio/contracts
Website	https://roseswap.io
Telegram	https://t.me/roseswapio
Twitter	https://twitter.com/roseswapio

Project Logo

Solidity Source Code On Rose Swap GitHub

https://github.com/roseswapio/contracts

Solidity Source Code On Oasis Network

❖ RoseRouter.sol

https://explorer.emerald.oasis.dev/address/0x018e712424baF2C80A6CD5FE6721FCaaE78533Ba/contracts

❖ RoseFactory.sol

https://explorer.emerald.oasis.dev/address/0x14593D7931ed5bB16C03b1F9a8ff6f1CeaBE6AAB/contracts

❖ RoseToken.sol

https://explorer.emerald.oasis.dev/address/0x0bfF36Be5cf671Fa973f8206483b6641A90CE7d0/contracts

❖ MasterChef.sol

https://explorer.emerald.oasis.dev/address/0x9Aa04E3C362eAb4E5a9f45c5897916834Aba41E8/contracts

❖ SafeMasterChefOwner.sol

https://explorer.emerald.oasis.dev/address/0x7bE7FB6b06041BEc1F65442Ae681851c15310Ba6/contracts

❖ Timelock.sol

https://explorer.emerald.oasis.dev/address/0x10f062972D964Bf965Bf78b2742625DA45615584/contracts

SHA-1 Hash

Solidity source codes are audited at hash #2a371abc5cf81574b6fd374e1fd80be5e65eed55

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Rose Swap. InterFi has 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	❖ Re-entrancy ❖ Unhandled Exceptions ❖ Transaction Order Dependency ❖ Integer Overflow ❖ Unrestricted Action ❖ Incorrect Inheritance Order ❖ Typographical Errors ❖ Requirement Violation
Source Code Review	❖ Ownership Takeover ❖ Gas Limit and Loops ❖ Deployment Consistency ❖ Repository Consistency ❖ Data Consistency ❖ Token Supply Manipulation
Functional Assessment	❖ Access Control and Authorization ❖ Operations Trail and Event Generation ❖ Assets Manipulation ❖ Liquidity Access

InterFi's Echelon Audit Standard

The aim of InterFi's "Echelon" standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
   ❖ Review of the specifications, sources, and instructions provided to InterFi to make sure we understand the size, scope, and functionality of the smart contract.
   ❖ Manual review of code, which is the process of reading source code line-byline to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:
   ❖ Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   ❖ Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

❖ Slither

❖ Consensys MythX, Mythril

❖ SWC Registry

❖ Solidity Coverage

❖ Open Zeppelin Code Analyzer

❖ Solidity Code Complier

InterFi's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract:

Vulnerable: A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level vulnerabilities could be exploited easily, and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity
! Medium	This level vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level vulnerabilities can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Smart Contract – Static Analysis

Symbol	Meaning
🛑	Function can be modified
💵	Function is payable
🔒	Function is locked
🔓	Function can be accessed
❗	Important functionality

```
| **Address** | Library |   |||
| └ | isContract | Internal 🔒 |   | |
| └ | sendValue | Internal 🔒 | 🛑 | |
| └ | functionCall | Internal 🔒 | 🛑 | |
| └ | functionCall | Internal 🔒 | 🛑 | |
| └ | functionCallWithValue | Internal 🔒 | 🛑 | |
| └ | functionCallWithValue | Internal 🔒 | 🛑 | |
| └ | _functionCallWithValue | Private 🔓 | 🛑 | |
||||||
| **IERC20** | Interface |   |||
| └ | totalSupply | External ❗ |   |NO❗ |
| └ | decimals | External ❗ |   |NO❗ |
| └ | symbol | External ❗ |   |NO❗ |
| └ | name | External ❗ |   |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | balanceOf | External ❗ |   |NO❗ |
| └ | transfer | External ❗ | 🛑  |NO❗ |
| └ | allowance | External ❗ |   |NO❗ |
| └ | approve | External ❗ | 🛑  |NO❗ |
| └ | transferFrom | External ❗ | 🛑  |NO❗ |
||||||
| **SafeERC20** | Library |   |||
| └ | safeTransfer | Internal 🔒 | 🛑 | |
| └ | safeTransferFrom | Internal 🔒 | 🛑  | |
| └ | safeApprove | Internal 🔒 | 🛑 | |
| └ | safeIncreaseAllowance | Internal 🔒 | 🛑 | |
| └ | safeDecreaseAllowance | Internal 🔒 | 🛑 | |
| └ | _callOptionalReturn | Private 🔓 | 🛑 | |
||||||
| **Context** | Implementation |   |||
| └ | <Constructor> | Internal 🔒 | 🛑 | |
```

```
| └ | _msgSender | Internal 🔒 |   | |
| └ | _msgData | Internal 🔒 |   | |
||||||
| **ERC20** | Implementation | Context, IERC20, Ownable |||
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | name | Public ❗ |   |NO❗ |
| └ | decimals | Public ❗ |   |NO❗ |
| └ | symbol | Public ❗ |   |NO❗ |
| └ | totalSupply | Public ❗ |   |NO❗ |
| └ | balanceOf | Public ❗ |   |NO❗ |
| └ | transfer | Public ❗ | 🔴  |NO❗ |
| └ | allowance | Public ❗ |   |NO❗ |
| └ | approve | Public ❗ | 🔴  |NO❗ |
| └ | transferFrom | Public ❗ | 🔴  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🔴  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🔴  |NO❗ |
| └ | mint | Public ❗ | 🔴  | onlyOwner |
| └ | _transfer | Internal 🔒 | 🔴  | |
| └ | _mint | Internal 🔒 | 🔴  | |
| └ | _burn | Internal 🔒 | 🔴  | |
| └ | _approve | Internal 🔒 | 🔴  | |
| └ | _burnFrom | Internal 🔒 | 🔴  | |
||||||
| **ISwapToken** | Interface | IERC20 |||
| └ | mint | External ❗ | 🔴  |NO❗ |
||||||
| **IMigratorChef** | Interface |  |||
| └ | migrate | External ❗ | 🔴  |NO❗ |
||||||
| **MasterChef** | Implementation | Ownable |||
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | updateMultiplier | Public ❗ | 🔴  | onlyOwner |
| └ | poolLength | External ❗ |   |NO❗ |
| └ | add | Public ❗ | 🔴  | onlyOwner |
| └ | set | Public ❗ | 🔴  | onlyOwner |
| └ | updateStakingPool | Internal 🔒 | 🔴  | |
| └ | setMigrator | Public ❗ | 🔴  | onlyOwner |
| └ | migrate | Public ❗ | 🔴  |NO❗ |
| └ | getMultiplier | Public ❗ |   |NO❗ |
| └ | pendingSwap | External ❗ |   |NO❗ |
| └ | massUpdatePools | Public ❗ | 🔴  |NO❗ |
| └ | updatePool | Public ❗ | 🔴  |NO❗ |
| └ | deposit | Public ❗ | 🔴  |NO❗ |
| └ | withdraw | Public ❗ | 🔴  |NO❗ |
| └ | enterStaking | Public ❗ | 🔴  |NO❗ |
| └ | leaveStaking | Public ❗ | 🔴  |NO❗ |
| └ | emergencyWithdraw | Public ❗ | 🔴  |NO❗ |
| └ | safeSwapTransfer | Internal 🔒 | 🔴  | |
| └ | dev | Public ❗ | 🔴  |NO❗ |
```

SOCIAL @interfinetwork WEB interfi.network

IRoseFactory	Interface				
└	feeTo	External ❗️			NO❗️
└	feeToSetter	External ❗️			NO❗️
└	getPair	External ❗️			NO❗️
└	allPairs	External ❗️			NO❗️
└	allPairsLength	External ❗️			NO❗️
└	createPair	External ❗️	🛑		NO❗️
└	setFeeTo	External ❗️	🛑		NO❗️
└	setFeeToSetter	External ❗️	🛑		NO❗️

IRosePair	Interface				
└	name	External ❗️			NO❗️
└	symbol	External ❗️			NO❗️
└	decimals	External ❗️			NO❗️
└	totalSupply	External ❗️			NO❗️
└	balanceOf	External ❗️			NO❗️
└	allowance	External ❗️			NO❗️
└	approve	External ❗️	🛑		NO❗️
└	transfer	External ❗️	🛑		NO❗️
└	transferFrom	External ❗️	🛑		NO❗️
└	DOMAIN_SEPARATOR	External ❗️			NO❗️
└	PERMIT_TYPEHASH	External ❗️			NO❗️
└	nonces	External ❗️			NO❗️
└	permit	External ❗️	🛑		NO❗️
└	MINIMUM_LIQUIDITY	External ❗️			NO❗️
└	factory	External ❗️			NO❗️
└	token0	External ❗️			NO❗️
└	token1	External ❗️			NO❗️
└	getReserves	External ❗️			NO❗️
└	price0CumulativeLast	External ❗️			NO❗️
└	price1CumulativeLast	External ❗️			NO❗️
└	kLast	External ❗️			NO❗️
└	mint	External ❗️	🛑		NO❗️
└	burn	External ❗️	🛑		NO❗️
└	swap	External ❗️	🛑		NO❗️
└	skim	External ❗️	🛑		NO❗️
└	sync	External ❗️	🛑		NO❗️
└	initialize	External ❗️	🛑		NO❗️

IRoseERC20	Interface				
└	name	External ❗️			NO❗️
└	symbol	External ❗️			NO❗️
└	decimals	External ❗️			NO❗️
└	totalSupply	External ❗️			NO❗️
└	balanceOf	External ❗️			NO❗️
└	allowance	External ❗️			NO❗️
└	approve	External ❗️	🛑		NO❗️
└	transfer	External ❗️	🛑		NO❗️
└	transferFrom	External ❗️	🛑		NO❗️

PAGE 11 | SMART CONTRACT SECURITY AUDIT OF ROSE SWAP

```
| └ | DOMAIN_SEPARATOR | External ❗ |   |NO❗ |
| └ | PERMIT_TYPEHASH | External ❗ |   |NO❗ |
| └ | nonces | External ❗ |   |NO❗ |
| └ | permit | External ❗ | 🔴  |NO❗ |
||||||
| **RoseERC20** | Implementation | IRoseERC20 |||
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | _mint | Internal 🔒 | 🔴  | |
| └ | _burn | Internal 🔒 | 🔴  | |
| └ | _approve | Private 🔐 | 🔴  | |
| └ | _transfer | Private 🔐 | 🔴  | |
| └ | approve | External ❗ | 🔴  |NO❗ |
| └ | transfer | External ❗ | 🔴  |NO❗ |
| └ | transferFrom | External ❗ | 🔴  |NO❗ |
| └ | permit | External ❗ | 🔴  |NO❗ |
||||||
| **Math** | Library |  |||
| └ | min | Internal 🔒 |   | |
| └ | sqrt | Internal 🔒 |   | |
||||||
| **UQ112x112** | Library |  |||
| └ | encode | Internal 🔒 |   | |
| └ | uqdiv | Internal 🔒 |   | |
||||||

| **IRoseCallee** | Interface |  |||
| └ | roseCall | External ❗ | 🔴  |NO❗ |
||||||
| **RosePair** | Implementation | IRosePair, RoseERC20 |||
| └ | getReserves | Public ❗ |   |NO❗ |
| └ | _safeTransfer | Private 🔐 | 🔴  | |
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | initialize | External ❗ | 🔴  |NO❗ |
| └ | _update | Private 🔐 | 🔴  | |
| └ | _mintFee | Private 🔐 | 🔴  | |
| └ | mint | External ❗ | 🔴  | lock |
| └ | burn | External ❗ | 🔴  | lock |
| └ | swap | External ❗ | 🔴  | lock |
| └ | skim | External ❗ | 🔴  | lock |
| └ | sync | External ❗ | 🔴  | lock |
||||||
| **RoseFactory** | Implementation | IRoseFactory |||
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | allPairsLength | External ❗ |   |NO❗ |
| └ | createPair | External ❗ | 🔴  |NO❗ |
| └ | setFeeTo | External ❗ | 🔴  |NO❗ |
| └ | setFeeToSetter | External ❗ | 🔴  |NO❗ |
||||||
| **TransferHelper** | Library |  |||
| └ | safeApprove | Internal 🔒 | 🔴  | |
```

```
| └ | safeTransfer | Internal 🔒 | 🔴 | |
| └ | safeTransferFrom | Internal 🔒 | 🔴 | |
| └ | safeTransferETH | Internal 🔒 | 🔴 | |
||||||
| **IRoseRouter01** | 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❗ |
||||||
| **IRoseRouter02** | Interface | IRoseRouter01 |||
| └ | removeLiquidityETHSupportingFeeOnTransferTokens | External ❗ | 🔴 |NO❗ |
| └ | removeLiquidityETHWithPermitSupportingFeeOnTransferTokens | External ❗ | 🔴 |NO❗ |
| └ | swapExactTokensForTokensSupportingFeeOnTransferTokens | External ❗ | 🔴 |NO❗ |
| └ | swapExactETHForTokensSupportingFeeOnTransferTokens | External ❗ | 💵 |NO❗ |
| └ | swapExactTokensForETHSupportingFeeOnTransferTokens | External ❗ | 🔴 |NO❗ |
||||||
| **IRoseFactory** | Interface | |||
| └ | feeTo | External ❗ | |NO❗ |
| └ | feeToSetter | External ❗ | |NO❗ |
| └ | getPair | External ❗ | |NO❗ |
| └ | allPairs | External ❗ | |NO❗ |
| └ | allPairsLength | External ❗ | |NO❗ |
| └ | createPair | External ❗ | 🔴 |NO❗ |
| └ | setFeeTo | External ❗ | 🔴 |NO❗ |
| └ | setFeeToSetter | External ❗ | 🔴 |NO❗ |
| └ | INIT_CODE_PAIR_HASH | External ❗ | |NO❗ |
||||||
| **IRosePair** | Interface | |||
| └ | name | External ❗ | |NO❗ |
| └ | symbol | External ❗ | |NO❗ |
| └ | decimals | External ❗ | |NO❗ |
| └ | totalSupply | External ❗ | |NO❗ |
| └ | balanceOf | External ❗ | |NO❗ |
| └ | allowance | External ❗ | |NO❗ |
```

```
| └ | approve | External ❗ | 🔴  |NO❗ |
| └ | transfer | External ❗ | 🔴  |NO❗ |
| └ | transferFrom | External ❗ | 🔴  |NO❗ |
| └ | DOMAIN_SEPARATOR | External ❗ |   |NO❗ |
| └ | PERMIT_TYPEHASH | External ❗ |   |NO❗ |
| └ | nonces | External ❗ |   |NO❗ |
| └ | permit | External ❗ | 🔴  |NO❗ |
| └ | MINIMUM_LIQUIDITY | External ❗ |   |NO❗ |
| └ | factory | External ❗ |   |NO❗ |
| └ | token0 | External ❗ |   |NO❗ |
| └ | token1 | External ❗ |   |NO❗ |
| └ | getReserves | External ❗ |   |NO❗ |
| └ | price0CumulativeLast | External ❗ |   |NO❗ |
| └ | price1CumulativeLast | External ❗ |   |NO❗ |
| └ | kLast | External ❗ |   |NO❗ |
| └ | mint | External ❗ | 🔴  |NO❗ |
| └ | burn | External ❗ | 🔴  |NO❗ |
| └ | swap | External ❗ | 🔴  |NO❗ |
| └ | skim | External ❗ | 🔴  |NO❗ |
| └ | sync | External ❗ | 🔴  |NO❗ |
| └ | initialize | External ❗ | 🔴  |NO❗ |
||||||
| **RoseLibrary** | Library |  |||
| └ | sortTokens | Internal 🔒 |   | |
| └ | pairFor | Internal 🔒 |   | |
| └ | getReserves | Internal 🔒 |   | |
| └ | quote | Internal 🔒 |   | |
| └ | getAmountOut | Internal 🔒 |   | |
| └ | getAmountIn | Internal 🔒 |   | |
| └ | getAmountsOut | Internal 🔒 |   | |
| └ | getAmountsIn | Internal 🔒 |   | |
||||||
| **BEP20** | Implementation | Context, IBEP20, Ownable |||
| └ | <Constructor> | Public ❗ | 🔴  |NO❗ |
| └ | getOwner | External ❗ |   |NO❗ |
| └ | name | Public ❗ |   |NO❗ |
| └ | decimals | Public ❗ |   |NO❗ |
| └ | symbol | Public ❗ |   |NO❗ |
| └ | totalSupply | Public ❗ |   |NO❗ |
| └ | balanceOf | Public ❗ |   |NO❗ |
| └ | transfer | Public ❗ | 🔴  |NO❗ |
| └ | allowance | Public ❗ |   |NO❗ |
| └ | approve | Public ❗ | 🔴  |NO❗ |
| └ | transferFrom | Public ❗ | 🔴  |NO❗ |
| └ | increaseAllowance | Public ❗ | 🔴  |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🔴  |NO❗ |
| └ | _transfer | Internal 🔒 | 🔴  | |
| └ | _mint | Internal 🔒 | 🔴  | |
| └ | _burn | Internal 🔒 | 🔴  | |
| └ | _approve | Internal 🔒 | 🔴  | |
```

└	_burnFrom	Internal 🔒	🛑		
RoseToken	Implementation	BEP20		
└	mint	Public ❗	🛑	onlyMinter
└	addMinter	Public ❗	🛑	onlyOwner
└	removeMinter	Public ❗	🛑	onlyOwner
└	burn	Public ❗	🛑	NO❗
└	burnFromDead	Public ❗	🛑	NO❗
└	delegates	External ❗		NO❗
└	delegate	External ❗	🛑	NO❗
└	delegateBySig	External ❗	🛑	NO❗
└	getCurrentVotes	External ❗		NO❗
└	getPriorVotes	External ❗		NO❗
└	_delegate	Internal 🔒	🛑	
└	_moveDelegates	Internal 🔒	🛑	
└	_writeCheckpoint	Internal 🔒	🛑	
└	safe32	Internal 🔒		
└	getChainId	Internal 🔒		
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
MasterChef	Interface			
└	add	External ❗	🛑	NO❗
└	set	External ❗	🛑	NO❗
└	updateMultiplier	External ❗	🛑	NO❗
SafeMasterChefOwner	Implementation	Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	add	External ❗	🛑	onlyOwner
└	set	External ❗	🛑	onlyOwner
└	updateMultiplier	External ❗	🛑	onlyOwner
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
└	min	Internal 🔒		

```
| └ | sqrt | Internal 🔒 |   | |
||||||
```

Timelock	Implementation			
└	<Constructor>	Public ❗	🔴 |NO❗	
└	<Receive Ether>	External ❗	💵 |NO❗	
└	setDelay	Public ❗	🔴 |NO❗	
└	acceptAdmin	Public ❗	🔴 |NO❗	
└	setPendingAdmin	Public ❗	🔴 |NO❗	
└	queueTransaction	Public ❗	🔴 |NO❗	
└	cancelTransaction	Public ❗	🔴 |NO❗	
└	executeTransaction	Public ❗	💵 |NO❗	
└	getBlockTimestamp	Internal 🔒		

InterFi

Smart Contract

Security Audit

Smart Contract – Software Analysis

Function Signatures

```
16279055  =>  isContract(address)
32749461  =>  getReserves(address,address,address)
39509351  =>  increaseAllowance(address,uint256)
74496190  =>  setMigrator(IMigratorChef)
24a084df  =>  sendValue(address,uint256)
a0b5ffb0  =>  functionCall(address,bytes)
241b5886  =>  functionCall(address,bytes,string)
2a011594  =>  functionCallWithValue(address,bytes,uint256)
d525ab8a  =>  functionCallWithValue(address,bytes,uint256,string)
36455e42  =>  _functionCallWithValue(address,bytes,uint256,string)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
f43f523a  =>  mod(uint256,uint256)
71af23e8  =>  mod(uint256,uint256,string)
7ae2b5c7  =>  min(uint256,uint256)
6d5433e6  =>  max(uint256,uint256)
677342ce  =>  sqrt(uint256)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
d0c407e1  =>  safeTransfer(IERC20,address,uint256)
5beae096  =>  safeTransferFrom(IERC20,address,address,uint256)
d6dcec8d  =>  safeApprove(IERC20,address,uint256)
390cc046  =>  safeIncreaseAllowance(IERC20,address,uint256)
5164ffed  =>  safeDecreaseAllowance(IERC20,address,uint256)
becc5a20  =>  _callOptionalReturn(IERC20,bytes)
119df25f  =>  _msgSender()
8b49d47e  =>  _msgData()
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
f2fde38b  =>  transferOwnership(address)
d29d44ee  =>  _transferOwnership(address)
a457c2d7  =>  decreaseAllowance(address,uint256)
a0712d68  =>  mint(uint256)
```

```
30e0789e  =>  _transfer(address,address,uint256)
4e6ec247  =>  _mint(address,uint256)
6161eb18  =>  _burn(address,uint256)
104e81ff  =>  _approve(address,address,uint256)
a22b35ce  =>  _burnFrom(address,uint256)
40c10f19  =>  mint(address,uint256)
6e08a0e3  =>  migrate(IERC20)
5ffe6146  =>  updateMultiplier(uint256)
081e3eda  =>  poolLength()
8b05fe68  =>  add(uint256,IERC20,bool)
64482f79  =>  set(uint256,uint256,bool)
9b9c4477  =>  updateStakingPool()
454b0608  =>  migrate(uint256)
8dbb1e3a  =>  getMultiplier(uint256,uint256)
47dc882f  =>  pendingSwap(uint256,address)
630b5ba1  =>  massUpdatePools()
51eb05a6  =>  updatePool(uint256)
e2bbb158  =>  deposit(uint256,uint256)
441a3e70  =>  withdraw(uint256,uint256)
41441d3b  =>  enterStaking(uint256)
1058d281  =>  leaveStaking(uint256)
5312ea8e  =>  emergencyWithdraw(uint256)
92a67b87  =>  safeSwapTransfer(address,uint256)
8d88a90e  =>  dev(address)
017e7e58  =>  feeTo()
094b7415  =>  feeToSetter()
e6a43905  =>  getPair(address,address)
1e3dd18b  =>  allPairs(uint256)
574f2ba3  =>  allPairsLength()
c9c65396  =>  createPair(address,address)
f46901ed  =>  setFeeTo(address)
a2e74af6  =>  setFeeToSetter(address)
3644e515  =>  DOMAIN_SEPARATOR()
30adf81f  =>  PERMIT_TYPEHASH()
7ecebe00  =>  nonces(address)
d505accf  =>  permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56  =>  MINIMUM_LIQUIDITY()
c45a0155  =>  factory()
544caa56  =>  sortTokens(address,address)
6d91c0e2  =>  pairFor(address,address,address)
bb7b9c76  =>  getAmountsOut(address,uint256,address[])
192128b2  =>  getAmountsIn(address,uint256,address[])
d0e30db0  =>  deposit()
2e1a7d4d  =>  withdraw(uint256)
6d7746bc  =>  _addLiquidity(address,address,uint256,uint256,uint256,uint256)
f5901d4d  =>  _swap(uint256[],address[],address)
d1c474e3  =>  _swapSupportingFeeOnTransferTokens(address[],address)
983b2d56  =>  addMinter(address)
3092afd5  =>  removeMinter(address)
42966c68  =>  burn(uint256)
```

```
11bb37b8  =>  burnFromDead()
587cde1e  =>  delegates(address)
5c19a95c  =>  delegate(address)
c3cda520  =>  delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32)
b4b5ea57  =>  getCurrentVotes(address)
782d6fe1  =>  getPriorVotes(address,uint256)
a28a42b3  =>  _delegate(address,address)
955f9fd8  =>  _moveDelegates(address,address,uint256)
ee59e77f  =>  _writeCheckpoint(address,uint32,uint256,uint256)
869d1f83  =>  safe32(uint256,string)
3408e470  =>  getChainId()
1eaaa045  =>  add(uint256,address,bool)
e177246e  =>  setDelay(uint256)
0e18b681  =>  acceptAdmin()
4dd18bf5  =>  setPendingAdmin(address)
3a66f901  =>  queueTransaction(address,uint256,string,bytes,uint256)
591fcdfe  =>  cancelTransaction(address,uint256,string,bytes,uint256)
0825f38f  =>  executeTransaction(address,uint256,string,bytes,uint256)
796b89b9  =>  getBlockTimestamp()
```

Inheritance Graph
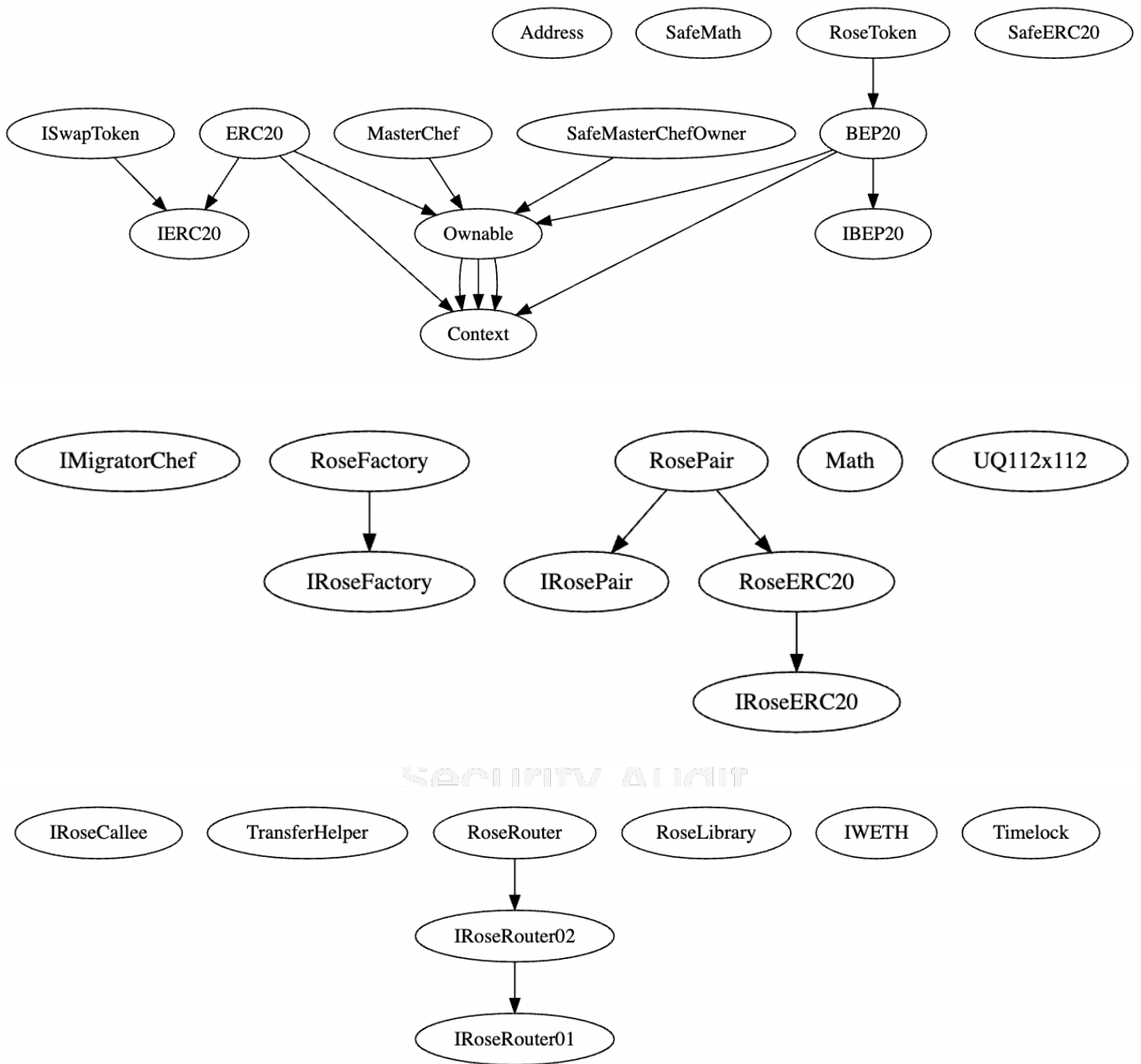
Smart Contract – Manual Analysis

❖ Rose Swap uses **MINT** to generate governance tokens. Mint can be called by Minter (MasterChef.sol) and Owner (Timelock.sol)

❖ Rose Swap's smart contract RoseFactory.sol has a low severity issue which may not create any functional vulnerability.

Expected identifier, got 'Payable'

"severity": NULL

❖ Rose Swap does not use "ReentrancyGuard" to prevent reentrant calls to a function. Reentrancy Guard is a contract module that helps prevent reentrant calls to a function. Inheriting from Reentrancy Guard makes the nonReentrant modifier available, which can be applied to functions to make sure there are no nested (reentrant) calls to them.

IMPORTANT: because control is transferred to `recipient`, care must be taken to not create reentrancy vulnerabilities. Consider using {ReentrancyGuard} or the https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-checks-effects-interactions-pattern [checks-effects-interactions pattern]

Smart Contract – SWC Attacks

| SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Low
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

SWC-119	Shadowing State Variables	Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	Passed
SWC-121	Missing Protection against Signature Replay Attacks	Passed
SWC-122	Lack of Proper Signature Verification	Passed
SWC-123	Requirement Violation	Passed
SWC-124	Write to Arbitrary Storage Location	Passed
SWC-125	Incorrect Inheritance Order	Passed
SWC-126	Insufficient Gas Griefing	Passed
SWC-127	Arbitrary Jump with Function Type Variable	Passed
SWC-128	DoS With Block Gas Limit	Passed
SWC-129	Typographical Error	Passed
SWC-130	Right-To-Left-Override control character (U+202E)	Passed
SWC-131	Presence of unused variables	Passed
SWC-132	Unexpected Ether balance	Passed
SWC-133	Hash Collisions With Multiple Variable Length Arguments	Passed
SWC-134	Message call with hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	Passed
SWC-136	Unencrypted Private Data On-Chain	Passed

Smart Contract - Risk Status & Radar Chart

Risk Severity	Status
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	2 low severity issues identified
Verified	54 functions and instances verified and checked
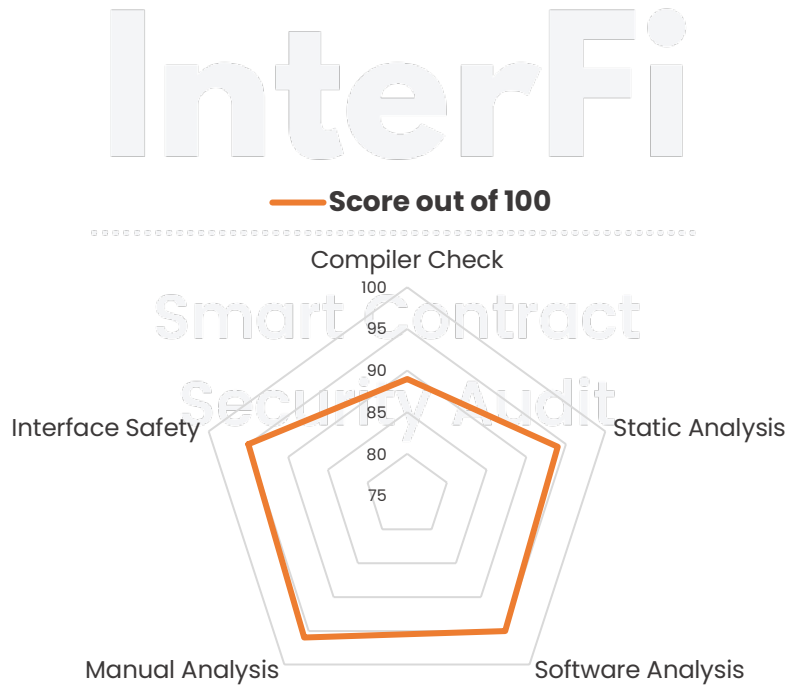
Auditor's Verdict

InterFi team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

- ❖ Rose Swap's smart contract source code has **LOW RISK SEVERITY**
- ❖ Rose Swap's smart contract has an **ACTIVE OWNERSHIP**

Note for stakeholders

- ❖ Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.

- ❖ Make sure that the project team's KYC/identity is verified by an independent firm, e.g., InterFi.

- ❖ Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in project's longevity. It is recommended to have multiple liquidity providers.

- ❖ Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.

- ❖ Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Important Disclaimer

InterFi Network provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project. **This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without InterFi's prior written consent.**

InterFi provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an enough assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. **Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, InterFi does not guarantee the explicit security of the audited smart contract.**

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided on this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

About InterFi Network

InterFi Network provides intelligent blockchain solutions. InterFi is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. **InterFi's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.**

InterFi is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ casual contributors. **InterFi provides manual, static, and automatic smart contract analysis, to ensure that project is checked against known attacks and potential vulnerabilities.**

To learn more, visit **https://interfi.network**

To view our audit portfolio, visit **https://github.com/interfinetwork**

To book an audit, message **https://t.me/interfiaudits**

@INTERFINETWORK

RELENTLESSLY SECURING THE PUBLIC BLOCKCHAIN | MADE IN CANADA 🇨🇦