



AUDIT REPORT

SecureWise

ROTTOLABS STAKING



Findings

Risk Classification	Description
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.
Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	A vulnerability that have informational character but is not effecting any of the code

Severity	Found	Pending	Resolved
High	0	1	0
Medium	0	0	0
Low	0	0	0
Informational	0	2	0
Total	0	3	0

Contents

02	Findings
04	Overview
05	Auditing Approach and Methodologies
06	Findings Summary
07	Function Privileges
08	Inheritance Graph
10	Manual Review
12	Disclaimer

Overview

Token Name: Rottolabs Staking

Language: Solidity

Contract Address: File

Network: Binance Smart Chain

KYC: KYC Done

Website: <https://rottpoken.com>

Twitter: <https://twitter.com/rotpolabs>

Telegram: <https://t.me/rotpotoken>

Report Date: July 20, 2023

Auditing Approach and Methodologies

SecureWise has performed starting with analyzing the code, issues, code quality, and libraries. Reviewed line-by-line by our team. Finding any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

Methodology

- Understanding the size, scope and functionality of your project's source code
- Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Testing and automated analysis of the Smart Contract to determine proper logic has been followed throughout the whole process
- Deploying the code on testnet using multiple live test
- Analyzing a program to determine the specific input that causes different parts of a program to execute its functions.
- Checking whether all the libraries used in the code are on the latest version.

Goals

Smart Contract System is secure, resilient and working according to the specifications and without any vulnerabilities.

Risk Classification

High: Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, of the contract and its functions. Must be fixed as soon as possible.

Medium: Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Must be fixed as soon as possible.




Low: Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.

Informational: A vulnerability that have informational character but is not effecting any of the code

Findings Summary

SecureWise has applied the automated and manual analysis of Smart Contract and were reviewed for common contract vulnerabilities and centralized exploits

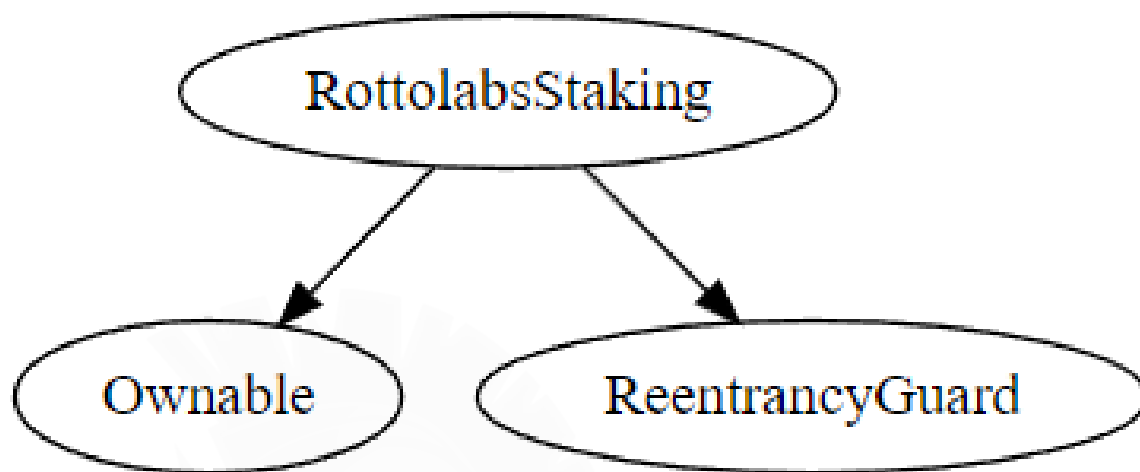
Findings

	Owner can create a blacklist and use it to block specific accounts
	Lack of parameter validation when creating plan
	Owner can withdraw the remaining coin funds when staking is over and they have no active users.

Function Privileges

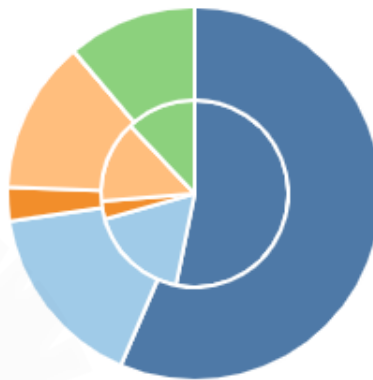
Contract	Type	Bases		
-----	-----	-----	-----	-----
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
RottolabsStaking	Implementation	Ownable, ReentrancyGuard		
L	<Constructor>	Public	!	NO !
L	stopContract	External	!	onlyWhenContractInactive onlyWhenNoStakers
L	startContract	External	!	onlyWhenContractInactive
L	generateInitialPlansPercent	Private	🔒	!
L	stake	External	!	nonReentrant notBlacklisted onlyWhenContractActive onlyWhenRottoApprove
L	getSelectedPlan	Internal	🔒	
L	getSelectedProduct	Internal	🔒	
L	unstakeTokens	External	!	nonReentrant onlyWhenContractActive onlyWhenRottoApprove
L	hasAllowance	Public	!	NO !
L	getStaker	Public	!	NO !
L	getPlans	Public	!	NO !
L	getPlansComplete	Public	!	NO !
L	_generatePlanCompleteInfo	Private	🔒	!
L	getLength	Public	!	NO !
L	getPlanLength	Internal	🔒	
L	removeStake	Private	🔒	!
L	calculatePercentage	Private	🔒	
L	calculateRewards	Public	!	NO !
L	createPlan	External	!	onlyOwner onlyWhenRottoApprove
L	deletePlan	External	!	onlyOwner onlyWhenRottoApprove
L	editServices	External	!	onlyOwner onlyWhenRottoApprove
L	updateContractAddress	External	!	onlyOwner onlyWhenRottoApprove
L	addToBlacklist	External	!	onlyOwner onlyWhenRottoApprove
L	removeFromBlacklist	External	!	onlyOwner onlyWhenRottoApprove
L	sendServices	Private	🔒	!
L	changeMaxStakers	External	!	onlyOwner onlyWhenRottoApprove
L	editTreasuryWallet	External	!	onlyOwner onlyWhenRottoApprove
L	editRottoApprove	External	!	onlyWhenIsRottoAdmin
L	withdrawalBalance	External	!	onlyOwner onlyWhenRottoApprove onlyWhenNoStakers

Inheritance Graph



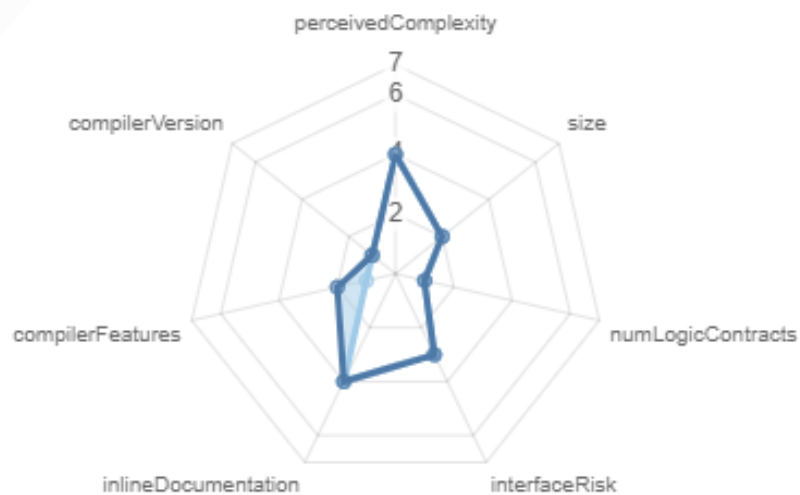
Source Lines

source comment single block mixed
empty todo blockEmpty



Risk

overall average



Manual Review

High Risk

Owner can create a blacklist and use it to block specific accounts

```
function addToBlacklist(address account↑)
  external
  onlyOwner
  onlyWhenRottoApprove {
    require(address(account↑) != address(0), "Invalid token address");
    blacklist[account↑] = true;
  }
```

```
function removeFromBlacklist(address account↑)
  external
  onlyOwner
  onlyWhenRottoApprove
{
  delete blacklist[account↑];
}
```

Description

Code does not include any mechanisms for verifying the legitimacy or justification of blacklisting accounts. Adding or removing accounts from the blacklist solely relies on the contract owner's authority. This lack of oversight could potentially lead to misuse or abuse of the blacklisting functionality.

Recommendation

Implementing a more robust and transparent mechanism for managing the blacklist. This could involve implementing a multi-signature approval process, utilizing an external oracle for account validation, or incorporating community governance mechanisms to prevent centralization of blacklist management.

Manual Review

Informational

Lack of parameter validation when creating plan

```
function createPlan(  
  uint256 _position↑,  
  uint128 _percentage↑,  
  uint256 _duration↑,  
  uint256 _minAmount↑,  
  uint256 _maxAmount↑,  
  bool _earlyWithdrawal↑  
) external onlyOwner onlyWhenRottoApprove {  
  require(  
    _percentage↑ >= MIN_PLAN_PERCENT && _percentage↑ <= MAX_PLAN_PERCENT,  
    "Percentage cannot be accepted"  
  );  
  require(  
    _minAmount↑ < _maxAmount↑,  
    "Minimum quantity cannot be equal to or greater than the maximum quantity"  
  );  
  require(  
    _duration↑ >= 1 minutes && _duration↑ <= 365 days,  
    "Duration cannot be less than 1 minute and more than 365 days"  
  );  
};
```

Description

There is no explicit check to ensure that the **_minAmount** parameter is greater than zero. This omission could potentially lead to issues where a plan with a minimum amount of zero could be created.

Recommendation

Update the function by adding a one more require statement to validate that the **_minAmount** parameter is greater than zero.

Disclaimer

SecureWise provides the smart contract audit of solidity. Audit and report are for informational purposes only and not, nor should be considered, as an endorsement to engage with, invest in, participate, provide an incentive, or disapprove, criticise, discourage, or purport to provide an opinion on any particular project or team.

This audit report doesn't provide any warranty or guarantee regarding the nature of the technology analysed. These reports, in no way, provide investment advice, nor should be used as investment advice of any sort. Investors must always do their own research and manage their risk.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and SecureWise and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) SecureWise owe no duty of care towards you or any other person, nor does SecureWise make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SecureWise hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SecureWise hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SecureWise, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.



AUDIT REPORT

SecureWise



securewise.org



t.me/securewisehub



twitter.com/securewiseAudit

