

S A L U S   S E C U R I T Y

N o v 2 0 2 5



# CODE SECURITY ASSESSMENT

P E E R T O K E N

# Overview

## Project Summary

- Name: PeerToken
- Platform: EVM-compatible chains
- Language: Solidity
- Address: [0x4a1EA452c598363188bcCC83FB4f69Dc14b54Acd](#)
- Audit Range: See [Appendix - 1](#)

# Project Dashboard

## Application Summary

Name	PeerToken
Version	v1
Type	Solidity
Dates	Nov 11 2025
Logs	Nov 11 2025

## Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	2
Total	3

## Contact

E-mail: support@salusec.io

# Risk Level Description

<b>High Risk</b>	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
<b>Medium Risk</b>	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
<b>Low Risk</b>	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
<b>Informational</b>	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

# Content

<b>Introduction</b>	<b>4</b>
1.1 About SALUS	4
1.2 Audit Breakdown	4
1.3 Disclaimer	4
<b>Findings</b>	<b>5</b>
2.1 Summary of Findings	5
2.2 Notable Findings	6
1. Centralization risk	6
2.3 Informational Findings	7
2. Missing two-step transfer ownership pattern	7
3. Lack of Security Contact Information for Responsible Disclosure	8
<b>Appendix</b>	<b>9</b>
Appendix 1 - Files in Scope	9

# Introduction

## 1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (<https://t.me/salusec>), Twitter ([https://twitter.com/salus\\_sec](https://twitter.com/salus_sec)), or Email ([support@salusec.io](mailto:support@salusec.io)).

## 1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

## 1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

# Findings

## 2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	<a href="#">Centralization risk</a>	Low	Centralization	Acknowledged
2	Missing two-step transfer ownership pattern	Informational	Business Logic	Acknowledged
3	Lack of Security Contact Information for Responsible Disclosure	Informational	Configuration	Acknowledged

## 2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Centralization risk	
Severity: Low	Category: Centralization
Target: - PeerToken.sol	

### Description

The `PeerToken` contract has privileged account `minter`. The `minter` has the authority mint PeerToken. If the `minter's` private key is compromised, it could cause the entire token ecosystem to collapse.

If the privileged accounts are plain EOA accounts, this can be worrisome and pose a risk to the other users.

### Recommendation

We recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

### Status

This issue has been acknowledged by the team.

## 2.3 Informational Findings

### 2. Missing two-step transfer ownership pattern

Severity: Informational

Category: Business logic

Target:

- PeerToken.sol

#### Description

The `PeerToken` contract inherits from the `Ownable` contract. This contract does not implement a two-step process for transferring ownership. Thus, ownership of the contract can easily be lost when making a mistake in transferring ownership.

#### Recommendation

Consider using the [Ownable2Step](#) contract from OpenZeppelin instead.

#### Status

This issue has been acknowledged by the team.

### **3. Lack of Security Contact Information for Responsible Disclosure**

Severity: Informational

Category: Configuration

Target:

- PeerToken.sol

#### **Description**

The contract `PeerToken` does not specify a security contact point. Including a designated security contact (such as an email address or ENS name) in the contract's NatSpec header facilitates responsible vulnerability disclosure. This makes it easier for external researchers to quickly reach the appropriate team in the event a vulnerability is identified, helping minimize the time window between discovery and mitigation. The Ethereum community has begun standardizing this practice using the `@custom:security-contact` tag, adopted by tools such as OpenZeppelin Wizard and ethereum-lists.

#### **Recommendation**

Consider adding a NatSpec comment at the top of the contract with a `@custom:security-contact` field pointing to the preferred disclosure channel.

#### **Status**

This issue has been acknowledged by the team.

# Appendix

## Appendix 1 - Files in Scope

This audit covered the following files at address

[0x4a1EA452c598363188bcCC83FB4f69Dc14b54Acd:](#)