SECURITY AUDIT REPORT

Canary - P2P

DATE
12 April 2025

PREPARED BY

0xTeam.
WEB3 AUDITS

info@0xTeam.Space
0xTeamSpace
OxTeamSpace

Contents

Revision History & Version Control

Version	Date	Description
1.0	07 April 2025	Initial Audit Report
2.0	10 April 2025	Second Audit Report
3.0	12 April 2025	Final Audit Report

0xTeam conducted a comprehensive Security Audit on the CanaryP2P to ensure the overall code quality, security, and correctness. The review focused on ensuring that the code functions as intended, identifying potential vulnerabilities, and safeguarding the integrity of CanaryP2P's operations against possible attacks.

Report Structure

The report is divided into two primary sections:

1. **Executive Summary** : Provides a high-level overview of the audit findings.
2. **Technical Analysis** : Offers a detailed examination of the Smart contracts code.

Note :
The analysis is static and exclusively focused on the Smart contracts code. The information provided in this report should be utilised to understand the security, quality, and expected behaviour of the code.

1.0 Disclaimer

This is a summary of our audit findings based on our analysis, following industry best practices as of the date of this report.However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. The audit focuses on Smart contracts coding practices and any issues found in the code, as detailed in this report. For a complete understanding of our analysis, you should read the full report. We have made every effort to conduct a thorough analysis, but it's important to note that you should not rely solely on this report and cannot make claims against us based on its contents. We strongly advise you to perform your own independent checks before making any decisions. Please read the disclaimer below for more information.

DISCLAIMER: By reading this report, you agree to the terms outlined in this disclaimer. If you do not agree, please stop reading immediately and delete any copies you have. This report is for informational purposes only and does not constitute investment advice. You should not rely on the report or its content, and 0xTeam and its affiliates (including all associated companies, employees, and representatives) are not responsible for any reliance on this report.The report is provided "as is" without any guarantees. 0xTeam excludes all warranties, conditions, or terms, including those implied by law, regarding quality, fitness for a purpose, and use of reasonable care. Except where prohibited by law, 0xTeam is not liable for any type of loss or damage, including direct, indirect, special, or consequential damages, arising from the use or inability to use this report. The findings are solely based on the Smart contracts code provided to us.

2.0 Executive Summary

2.1 Overview

0xTeam has meticulously audited the CanaryP2P Smart contracts project from 14 March 2025 to 27 March 2025. The primary objective of this audit was to assess the security, functionality, and reliability of the CanaryP2P's before their deployment on the blockchain. The audit focused on identifying potential vulnerabilities, evaluating the contract's adherence to best practices, and providing recommendations to mitigate any identified risks. The comprehensive analysis conducted during this period ensures that the CanaryP2P is robust and secure, offering a reliable environment for its users.

2.2 Scope

The scope of this audit involved a thorough analysis of the CanaryP2P Smart contracts, focusing on evaluating its quality, rigorously assessing its security, and carefully verifying the correctness of the code to ensure it functions as intended without any vulnerabilities.

Files in Examination:

Language	Solidity
In-Scope	• contracts/p2p/P2pFactory.sol • contracts/p2p/P2pV2.sol • contracts/p2p/P2pV1.sol • contracts/p2p/interfaces/IP2p.sol
Fixed Review Commit Hash	45473ed14b7277f3f1aa2dec90415cbe9cb4953e

OUT-OF-SCOPE: External Smart contracts code, other imported code.

2.3 Audit Summary

Name	Verified	Audited	Vulnerabilities
CanaryP2P	Yes	Yes	Refer Section 5.0

2.4 Summary of Findings

ID	Title	Severity	Fixed
[L-01]	Freeze Initializers in Constructor to Secure the Implementation Contract	LOW	✔

2.5 Vulnerability Summary

High	Medium	Low	Informational
0	0	1	0

High Medium Low Informational

2.6 Recommendation Summary

Severity

Issues

	High	Medium	Low	Informational
Open	0	0	1	0
Resolved			1	
Acknowledged				
Partially Resolved				

- **Open**: Unresolved security vulnerabilities requiring resolution.
- **Resolved**: Previously identified vulnerabilities that have been fixed.
- **Acknowledged**: Identified vulnerabilities noted but not yet resolved.
- **Partially Resolved**: Risks mitigated but not fully resolved.

3.0 Checked Vulnerabilities

We examined Smart contracts for widely recognized and specific vulnerabilities. Below are some of the common vulnerabilities considered.

Category	Check Items
Source Code Review	→ Reentrancy Vulnerabilities → Ownership Control → Time-Based Dependencies → Gas Usage in Loops → Transaction Sequence Dependencies → Style Guide Compliance → EIP Standard Compliance → External Call Verification → Mathematical Checks → Type Safety → Visibility Settings → Deployment Accuracy → Repository Consistency
Functional Testing	→ Business Logic Validation → Feature Verification → Access Control and Authorization → Escrow Security → Token Supply Management → Asset Protection → User Balance Integrity → Data Reliability → Emergency Shutdown Mechanism

4.0 Techniques , Methods & Tools Used

The following techniques, methods, and tools were used to review all the smart contracts

- **Structural Analysis:**
  EThis involves examining the overall design and architecture of the smart contract. We ensure that the contract is logically organised, scalable, and follows industry best practices. This step is crucial for identifying potential structural issues that could lead to vulnerabilities or maintenance challenges in the future.

- **Static Analysis:**
  Static analysis is conducted using automated tools to scan the contract's codebase for common vulnerabilities and security risks without executing the code. This process helps identify issues such as reentrancy, arithmetic errors, and potential denial-of-service (DOS) vulnerabilities early on, allowing for quick remediation.

- **Code Review / Manual Analysis:**
  A manual, in-depth review of the smart contract's code is performed to verify the logic and ensure it matches the intended functionality as described in the project's documentation. During this phase, we also confirm the findings from the static analysis and check for any additional issues that may not have been detected by automated tools.

- **Dynamic Analysis:**
  Dynamic analysis involves executing the smart contract in various controlled environments to observe its behaviour under different conditions. This step includes running comprehensive test cases, performing unit tests, and monitoring gas consumption to ensure the contract operates efficiently and securely in real-world scenarios.

- **Tools and Platforms Used for Audit:**
  Utilising tools such as Remix , Slither, Aderyn, Solhint for static analysis, and platforms like Hardhat and Foundry for dynamic testing and simulation.

Note: The following values for "Severity" mean:

- High: Direct and severe impact on the funds or the main functionality of the protocol.
- Medium: Indirect impact on the funds or the protocol's functionality.
- Low: Minimal impact on the funds or the protocol's main functionality.
- Informational: Suggestions related to good coding practices and gas efficiency.

5.0 Technical Analysis

Low

[L-01] Freeze Initializers in Constructor to Secure the Implementation Contract

Severity

Low

Issue Description

When using `UUPSUpgradeable`, ensure the implementation contract cannot be initialized by others. Call `_freezeInitializers()` inside the constructor to lock initialization.

This prevents accidental or malicious setup and secures the contract.

Recommendation

Call `_freezeInitializers()` inside the constructor to disable initialization on the implementation contract.

```
constructor() {
    _freezeInitializers();
}
```

Status

Fixed

6.0 Auditing Approach and Methodologies Applied

The Solidity smart contract was audited using a comprehensive approach to ensure the highest level of security and reliability. Careful attention was given to the following key areas to ensure the overall quality of the code:

- **Code quality and structure**: We conducted a detailed review of the codebase to identify any potential issues related to code structure, readability, and maintainability. This included analysing the overall architecture of the Solidity smart contract and reviewing the code to ensure it follows best practices and coding standards.
- **Security vulnerabilities**: Our team used manual techniques to identify any potential security vulnerabilities that could be exploited by attackers. This involved a thorough analysis of the code to identify any potential weaknesses, such as buffer overflows, injection vulnerabilities, signatures, and deprecated functions.
- **Documentation and comments**: Our team reviewed the code documentation and comments to ensure they accurately describe the code's intended behaviour and logic. This helps developers to better understand the codebase and make modifications without introducing new issues.
- **Compliance with best practices**: We checked that the code follows best practices and coding standards that are recommended by the Solidity community and industry experts. This ensures that the Solidity smart contract is secure, reliable, and efficient.

Our audit team followed OWASP and Ethereum (Solidity) community security guidelines for this audit. As a result, we were able to identify potential issues and provide recommendations to improve the smart contract's security and performance.

Throughout the audit of the smart contracts, our team placed great emphasis on ensuring the overall quality of the code and the use of industry best practices. We meticulously reviewed the codebase to ensure that it was thoroughly documented and that all comments and logic aligned with the intended behaviour. Our approach to the audit was comprehensive, methodical, and aimed at ensuring that the smart contract was secure, reliable, and optimised for performance.

6.1 Code Review / Manual Analysis

Our team conducted a manual analysis of the Solidity smart contracts to identify new vulnerabilities or to verify vulnerabilities found during static and manual analysis. We carefully analysed every line of code and made sure that all instructions provided during the onboarding phase were followed. Through our manual analysis, we were able to identify potential vulnerabilities that may have been missed by automated tools and ensure that the smart contract was secure and reliable.

6.2 Tools Used for Audit

In the course of our audit, we leveraged a suite of tools to bolster the security and performance of our program. While our team drew on their expertise and industry best practices, we also integrated various tools into our development environment. Noteworthy among them are Remix,Slither, Aderyn, Solhint for Static Analysis and Hardhat & Foundry for Dynamic Analysis. This holistic approach ensures a thorough analysis, uncovering potential issues that automated tools alone might overlook. 0xTeam takes pride in utilising these tools, which significantly contribute to the quality, security, and maintainability of our codebase

7.0 Limitations on Disclosure and Use of this Report

This report contains information concerning potential details of the CanaryP2P Project and methods for exploiting them. 0xTeam recommends that special precautions be taken to protect the confidentiality of both this document and the information contained herein. Security Assessment is an uncertain process, based on past experiences, currently available information, and known threats. All information security systems, which by their nature are dependent on human beings, are vulnerable to some degree. Therefore, while 0xTeam considers the major security vulnerabilities of the analysed systems to have been identified, there can be no assurance that any exercise of this nature will identify all possible vulnerabilities or propose exhaustive and operationally viable recommendations to mitigate those exposures. In addition, the analysis set forth herein is based on the technologies and known threats as of the date of this report. As technologies and risks change over time, the vulnerabilities associated with the operation of the CanaryP2P Smart contracts Code Base described in this report, as well as the actions necessary to reduce the exposure to such vulnerabilities, will also change. 0xTeam makes no undertaking to supplement or update this report based on changed circumstances or facts of which 0xTeam becomes aware after the date hereof, absent a specific written agreement to perform the supplemental or updated analysis. This report may recommend that 0xTeam use certain software or hardware products manufactured or maintained by other vendors. 0xTeam bases these recommendations upon its prior experience with the capabilities of those products. Nonetheless, 0xTeam does not and cannot warrant that a particular product will work as advertised by the vendor, nor that it will operate in the manner intended. This report was prepared by 0xTeam for the exclusive benefit of CanaryP2P and is proprietary information. The Non-Disclosure Agreement (NDA) in effect between 0xTeam and CanaryP2P governs the disclosure of this report to all other parties including product vendors and suppliers.