SOURCEHAT

# Project Pi EarlyStaking

## Smart Contract Audit Report

## EXECUTIVE SUMMARY

This report presents the outcomes of our collaborative engagement with the Project Pi team, focusing on the comprehensive evaluation of the EarlyStaking2 and Storage contracts.

Our team conducted an initial security assessment from **February 1st** to **February 5th, 2024**. On **February 7th,** our team amended this report to reflect changes made to the contracts to resolve the findings we had identified during our initial review.

Pi Staking, a staking protocol designed for PulseChain, aims to allow node operators to initiate their operations swiftly and affordably through the utilization of the PPY token.

## AUDIT SCOPE

| Name | Source Code | Visualized |
|---|---|---|
| **EarlyStaking2** | Provided by the team. | Inheritance Chart. Function Graph. |
| **Storage** | Provided by the team. | Inheritance Chart. Function Graph. |

## AUDIT FINDINGS

All findings have been resolved, though some centralized aspects are present.

**Finding #1 EarlyStaking2** High Resolved

**Description:** The contract allows users to stake PLS to earn rewards. Each user's balance is tracked internally by the contract to ensure users cannot withdraw more PLS than they actually staked. Users may also manually claim their rewards. When claiming rewards, users' staked balance is deposited into an external stPLS contract. However, the user's balance is not decremented by the amount being deposited into the stPLS contract.

**Risk/Impact:** Users will potentially be able to withdraw another user's deposited PLS. A user can stake PLS and then claim their rewards. The PLS the user originally deposited will be sent to the stPLS contract. The same user can then withdraw and if there is sufficient PLS receive their full staked amount despite the PLS being sent to another contract during the rewards claim.

**Recommendation:** The team should set a user's staked balance to 0 after sending their deposited PLS to the stPLS contract in the claimRewards() function.

**Resolution:** The team has implemented the above recommendation.

### Finding #2 EarlyStaking2

**Informational** **Resolved**

**Description:** The contract grants an approval to itself in the claimRewards() function. The contract subsequently uses the transferFrom() function with its own address as the sedner.

**Recommendation:** The team should remove the approval and instead use the transfer() function.

**Resolution:** The team has implemented the above recommendation.

## SYSTEM OVERVIEW

**STAKING**

The EarlyStaking2 contract allows users to deposit PLS in order to earn rewards. Any user may stake any amount of PLS at any time. The contract also maintains the peak amount of PLS each user has staked at any given time.

**UNSTAKING**

Users may unstake their PLS from the EarlyStaking2 contract at any time.

**REWARDS**

The contract pays staking users rewards in the form of PPY tokens. Rewards are distributed at the "reward rate" per second. Users then earn rewards based on the amount of tokens they have staked and the amount of time they have staked.

Users may claim their rewards after the reward end timestamp has passed. If the amount the user has staked is not currently the peak amount they have staked they will forfeit some of their rewards. The amount of rewards the user will receive is proportional to the ratio of the user's current amount staked and their peak amount staked. Any forfeited rewards will be sent to the 0xdead address. The remaining rewards are transferred to the user. Additionally, the users currently staked balance will be "deposited" in the stPLS contract when claiming rewards. The stPLS contract was outside the scope of this audit so we are unable to give an assessment in regard to security.

**STORAGE**

The Storage contract is used to maintain the state of the EarlyStaking2 contract. The Guardian address may set any value in the contract at any time. The Guardian may also set any number of Network contracts. These contracts may also set any value in the contract at any time.

The Guardian address may propose a new Guardian address at any time. Once the new Guardian address has been proposed the new Guardian must accept the role for the address to be updated.

## VULNERABILITY ANALYSIS

| Vulnerability Category | Notes | Result |
|---|---|---|
| Arbitrary Jump/Storage Write | N/A | PASS |
| Centralization of Control | The Storage contract's Guardian address may update any value. | WARNING |
| Compiler Issues | N/A | PASS |

| | | |
|---|---|---|
| Delegate Call to Untrusted Contract | N/A | PASS |
| Dependence on Predictable Variables | N/A | PASS |
| Ether/Token Theft | N/A | PASS |
| Flash Loans | N/A | PASS |
| Front Running | N/A | PASS |
| Improper Events | N/A | PASS |
| Improper Authorization Scheme | N/A | PASS |
| Integer Over/Underflow | N/A | PASS |
| Logical Issues | N/A | PASS |
| Oracle Issues | N/A | PASS |
| Outdated Compiler Version | N/A | PASS |
| Race Conditions | N/A | PASS |
| Reentrancy | N/A | PASS |
| Signature Issues | N/A | PASS |
| Sybil Attack | N/A | PASS |
| Unbounded Loops | N/A | PASS |
| Unused Code | N/A | PASS |
| Overall Contract Safety | | PASS |

## *ABOUT SOURCEHAT*

SourceHat (formerly Solidity Finance - founded in 2020) has quickly grown to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1700+ solidity smart contract audits covering all major project types and protocols, securing a total of over $50 billion U.S. dollars in on-chain value!
Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

## *WHAT IS A SOURCEHAT AUDIT?*

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *SourceHat Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

## *HOW DO I INTERPRET THE FINDINGS?*

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.

- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.

GO HOME