33AUDITS & CO.

2Phux Audit Report

Introduction

A time-boxed security review of the protocol was done by 33Audit & Company, focusing on the security aspects of the smart contracts.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource, and expertise-bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any vulnerabilities. Subsequent security reviews, bug bounty programs, and on-chain monitoring are recommended.

About 33 Audits & Company

33Audits LLC is an independent smart contract security researcher company and development group. We conduct audits a as a group of independent auditors with various experience and backgrounds. We have conducted over 15 audits with dozens of vulnerabilities found and are experienced in building and auditing smart contracts. We have over 4 years of Smart Contract development with a focus in Solidity, Rust and Move. Check our previous work here or reach out on X @solidityauditor.

About 2PHUX

This audit is for minor changes that were made to 2PHUX since the fork of Aura finance. These changes are made in the contracts listed under the scope above. The other contracts are out of scope and were not audited by 33Audits & Co.

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Impact - The technical, economic, and reputation damage from a successful attack

Likelihood - The chance that a particular vulnerability gets discovered and exploited

Severity - The overall criticality of the risk

Informational - Findings in this category are recommended changes for improving the structure, usability, and overall effectiveness of the system.

Security Assessment Summary

***review commit hash* - 914e4143b7f70b8bea208ec9c1e1f0c364b08e73**

No fixes implemented.

Scope

The following smart contracts were in the scope of the audit:

- `contracts/core/Aura.sol`
- `contracts/convex-contracts/ConvexMasterChef.sol`
- `contracts/convex-contracts/Booster.sol`

Findings Summary

ID	Title	Severity	Status
[H-01]	Setting Emissions to the wrong value could lead to loss of funds for protocol	High	
[L-01]	Missing setter function for the `minter` address	Low	-
[L-02]	Use of outdated OpenZeppelin dependencies	Low	-
[L-02]	Use of outdated OpenZeppelin dependencies	Low	-
[NC-01]	`minterMinted` state variable can be gas optimized	Low	-
[NC-02]	Discourage use of `console.log` in production	Low	-
[NC-03]	Calculating blocks per year can be tricky due to leap year	Low	-

Detailed Findings

[H-1] - Setting Emissions to the wrong value could lead to loss of funds for protocol

Links:

https://github.com/RedDuck-Software/prime2phux-contracts/blob/914e4143b7f70b8bea208ec9c1e1f0c364b08e73/contracts/convex-contracts/contracts/contracts/ConvexMasterChef.sol#L55

Severity:

Medium

Description:

Its unclear from the information we were given what the exact value of `rewardPerBlock` will be. From the details that we were given it looks like the value that will be passed is `1e27` this could lead to a loss of funds as the token supply is `1e27` and so all of the tokens would be rewarded to the first depositor in the first block.

Impact:

All of the tokens will be rewarded to the first rewarded.

Recommendations:

It was stated in the text file that was sent to us that the amount being passed for `rewardPerBlock` would actually be `1e27 / 4 years`. Still it is recommended to find the average amount of blocks per year, multiple that by four and then divide 1e27 by the following number. You can reference this link here to find out more information about the amount of blocks that happen per year.

Resolution:

[L-1] Missing setter function for the `minter` address

Links:

https://github.com/RedDuck-Software/prime2phux-contracts/blob/914e4143b7f70b8bea208ec9c1e1f0c364b08e73/contracts/core/Aura.sol#L25

Severity:

Low

Description:

The `minter` address is set only once in the `init` function and cannot be updated thereafter. If the minter is compromised or the private keys are lost, there's no way to update the `minter` address, which can lead to a compromised accounts.

Impact:

A compromised `minter` address can mint unlimited Aura Tokens.

Recommendations:

Add a setter function with proper access control to allow updating the `minter` address in case of compromise or loss of private keys.

```
    function setMinter(address _minter) public onlyOwner {
        minter = _minter;
    }
```

[L-2] Use of outdated OpenZeppelin dependencies

Links:

https://github.com/RedDuck-Software/prime2phux-contracts/blob/914e4143b7f70b8bea208ec9c1e1f0c364b08e73/package.json#L58

Severity:

Low

Description:

The current project utilizes a forked version built on top of OpenZeppelin contract version `v4.4.2`, released around January 2022. Since then, several issues have been discovered in these OpenZeppelin contracts. As a best practice, it is advisable to use the latest stable version of OpenZeppelin contracts. However, considering this is a forked version, it's important to be cautious as even small changes can impact the contract's workflow and some component of the code may stop working.

Impact:

The contracts may be exploited due to vulnerabilities present in the outdated OpenZeppelin contracts.

POC:

https://security.snyk.io/package/npm/@openzeppelin%2Fcontracts

Recommendations:

Upgrade to the latest OpenZeppelin releases.

Resolution:

[NC-1] - `minterMinted` state variable can be gas optimized

Links:

https://github.com/RedDuck-Software/prime2phux-contracts/blob/914e4143b7f70b8bea208ec9c1e1f0c364b08e73/contracts/core/Aura.sol#L26

Severity:

Informational

Description:

The `minterMinted` variable is initialized to `type(uint256).max` in the constructor, but it is later set to 0 in the `init` function. Initializing it directly to zero would be a more efficient approach.

Recommendations:

Initialize the value of `minterMinted` to zero directly.

Resolution:

[NC-2] Discourage use of `console.log` in production

Links:

https://github.com/RedDuck-Software/prime2phux-contracts/blob/914e4143b7f70b8bea208ec9c1e1f0c364b08e73/contracts/convex-contracts/contracts/contracts/Booster.sol#L10

Severity:

Informational

Description:

The `console.log` statements are generally used by developers during the debugging phase to understand the flow and verify certain aspects of the code. However, as a common security and coding standard, using these in production environments is discouraged.

Recommendations:

Remove `console.log` statements from the production codebase.

Resolution:

[NC-3] Calculating blocks per year can be tricky due to leap year

Links:

https://github.com/RedDuck-Software/prime2phux-contracts/blob/914e4143b7f70b8bea208ec9c1e1f0c364b08e73/contracts/convex-contracts/contracts/contracts/ConvexMasterChef.sol#L55

Severity:

Non-critcal

Description:

Calculating the reward per block over a period of four years can be challenging due to the inclusion of leap years. The typical calculation of multiplying 365 days by 4 does not account for the extra day in a leap year. To accurately calculate the total number of days over four years, one should multiply 365 by 4 and then add 1 for the leap year, resulting in 1,461 days. Given that the number of blocks per day is fixed at 7,166, the total number of blocks can then be calculated by multiplying the total number of days by the number of blocks per day.

Impact:

Incorrect calculations can lead to discrepancies in the expected rewards per block over a four-year cycle, affecting projections and operations that depend on these figures.

POC:

To illustrate the calculation issue:

- Assume the non-leap year calculation: 365 days/year * 4 years = 1,460 days.
- Correct calculation including leap year: (365 days/year * 4 years) + 1 day = 1,461 days.
- Multiply the correct number of days by the fixed blocks per day: 1,461 days * 7,166 blocks/day = 10,468,126 blocks.

Recommendations:

Ensure the accuracy of block-related calculations

Resolution: