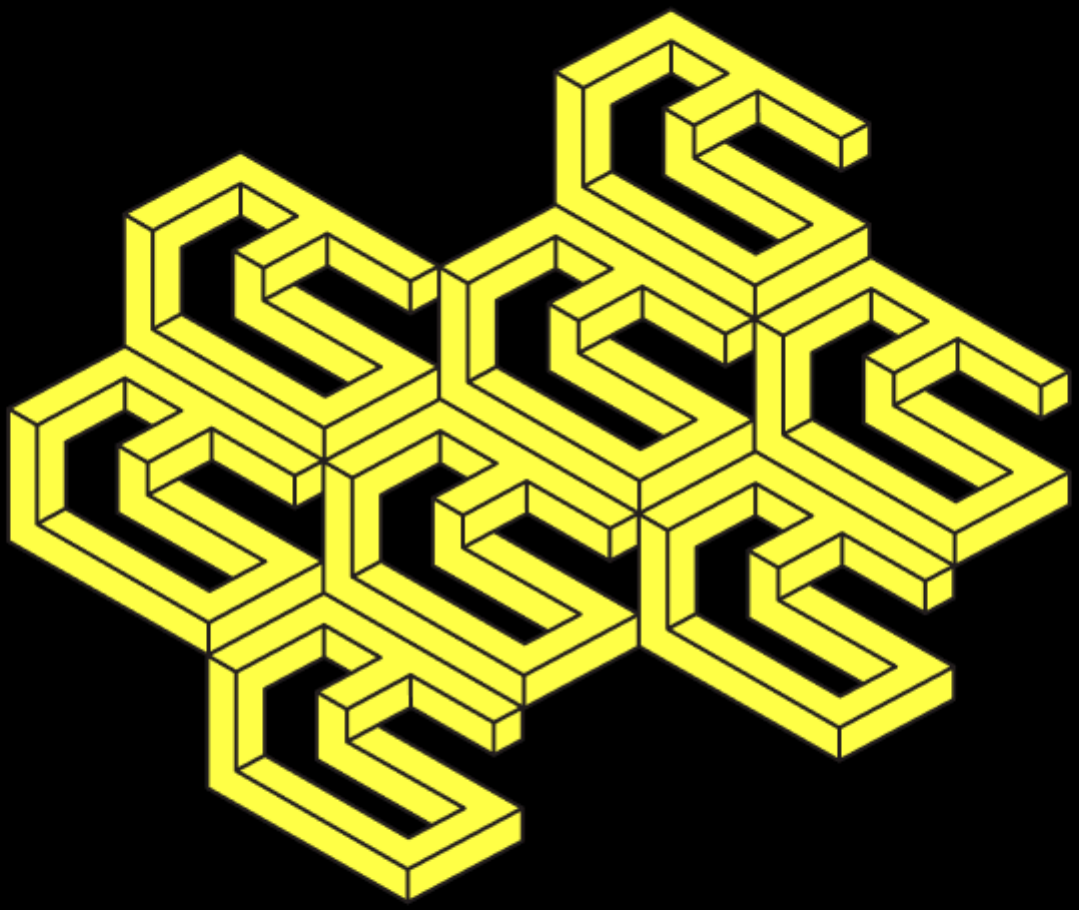

Propy NFT Staking V2 Smart Contract Review | May 2024

by ChainSafe Systems | May 2024

Table of contents

1. Introduction

Date	Auditor(s)
May 2024	Oleksii Matiiasevych, Anderson Lee, Tanya Bushenyova

Propy requested **ChainSafe Systems** to perform a review of the NFT staking smart contracts (version 2). The contracts can be identified by the following git commit hash: `7e5cac32f9451d5ac67d802562262a5bfbf249c9`

There is 1 contract in scope (`PRONFTStakingV2.sol`).

After the initial review, Propy team applied a number of updates which can be identified by the following git commit hash: `b64259e570914dff2f33da10bfb6db727b07195e`

Additional verification was performed after that.

Defining Severity

Each finding is assigned a severity level.

Note	Notes are informational in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
Optimization	Optimizations are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
Minor	Minor issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
Major	Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
Critical	Critical issues are directly exploitable security vulnerabilities that need to be fixed.

Referencing updated code

Resolved	The finding has been acknowledged and the team has since updated the code.
Rejected	The team dismissed this finding and no changes will be made.

Disclaimer

The review makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about the fitness of the contracts for any specific purpose, or their bug free status.

2. Executive Summary

All the initially identified issues were fixed and are not present in the final version of the contract.

There are **no** known compiler bugs for the specified compiler version (0.8.4), that might affect the contracts' logic.

There were **0 critical**, **0 major**, **0 minor**, 8 informational/optimization issues identified in the initial version of the contracts. All the issues found in the contract were not present in the final version of the contracts. They are described below for historical purposes. We enjoyed working with the Propy team, and liked how open and engaged they were in the discussion and improvement process throughout the review.

3. Critical Bugs and Vulnerabilities

No critical issues were identified.

4. Line-by-line review

contracts/PRONFTStakingV2.sol

L17-19 Optimization Resolved

`baseProToken`, `propyNFT`, and `ogNFT` could be set immutable.

L125 Optimization Resolved

```
uint256 stakingPowerMinted = amount;
```

In the `enter()` function `stakingPowerMinted` is always equal to `amount` so the `stakingPowerMinted` variable can be removed.

L132-136 Optimization Resolved

```
if(_tokenAddress == address(propyNFT)) {
    propyNFT.transferFrom(msg.sender, address(this), nftId);
} else {
    ogNFT.transferFrom(msg.sender, address(this), nftId);
}
```

There is no need to validate if `_tokenAddress` is equal to `propyNFT`. It would be enough just to call `transferFrom()` function of the `_tokenAddress` contract.

L152 Optimization Resolved

In the `enter()` function calling `updateCumulativeRewardsPerShare()` twice is excessive and could be optimized. Only `totalTrackedPRO` should be updated at the end of the function (which is already updated on line 143).

L180-181 Optimization Resolved

```
uint256 stakerCumulativeRewardsPerShare =
stakerToCumulativeRewardsPerShare[msg.sender];
uint256 rewards = (cumulativeRewardsPerShare -
stakerCumulativeRewardsPerShare) * share / 1e8;
```

In the `leave()` function `stakerToCumulativeRewardsPerShare[msg.sender]` and `cumulativeRewardsPerShare` are read from storage at each iteration of the for loop. They could be cached in a local variable before for loop.

L198 Optimization Resolved

In the `leave()` function calling `updateCumulativeRewardsPerShare()` twice is excessive and could be optimized. Only `totalTrackedPRO` should be updated at the end of the function (which is already updated on line 189).

L202-207 Optimization Resolved

```
if (totalSupply() > 0) {
    uint256 currentBalance = baseProToken.balanceOf(address(this));
if (currentBalance > totalTrackedPRO) {
    uint256 newRewards = currentBalance - totalTrackedPRO;
    cumulativeRewardsPerShare += (newRewards * 1e8) / totalSupply();
}
```

In the `updateCumulativeRewardsPerShare()` function `totalSupply()` and `totalTrackedPRO` are read from storage twice. They could be read once and stored in local variables.

L202-211 Optimization Resolved

```
if (totalSupply() > 0) {
    uint256 currentBalance = baseProToken.balanceOf(address(this));
    if (currentBalance > totalTrackedPRO) {
        uint256 newRewards = currentBalance - totalTrackedPRO;
        cumulativeRewardsPerShare += (newRewards * 1e8) / totalSupply();
    }
    totalTrackedPRO = currentBalance;
} else {
    totalTrackedPRO = baseProToken.balanceOf(address(this));
}
```

In the `updateCumulativeRewardsPerShare()` function `currentBalance` could be set in the first line of the function and the `else` case could be removed. Instead, `totalTrackedPRO` could just be set equal to `currentBalance` outside of the `if` condition.