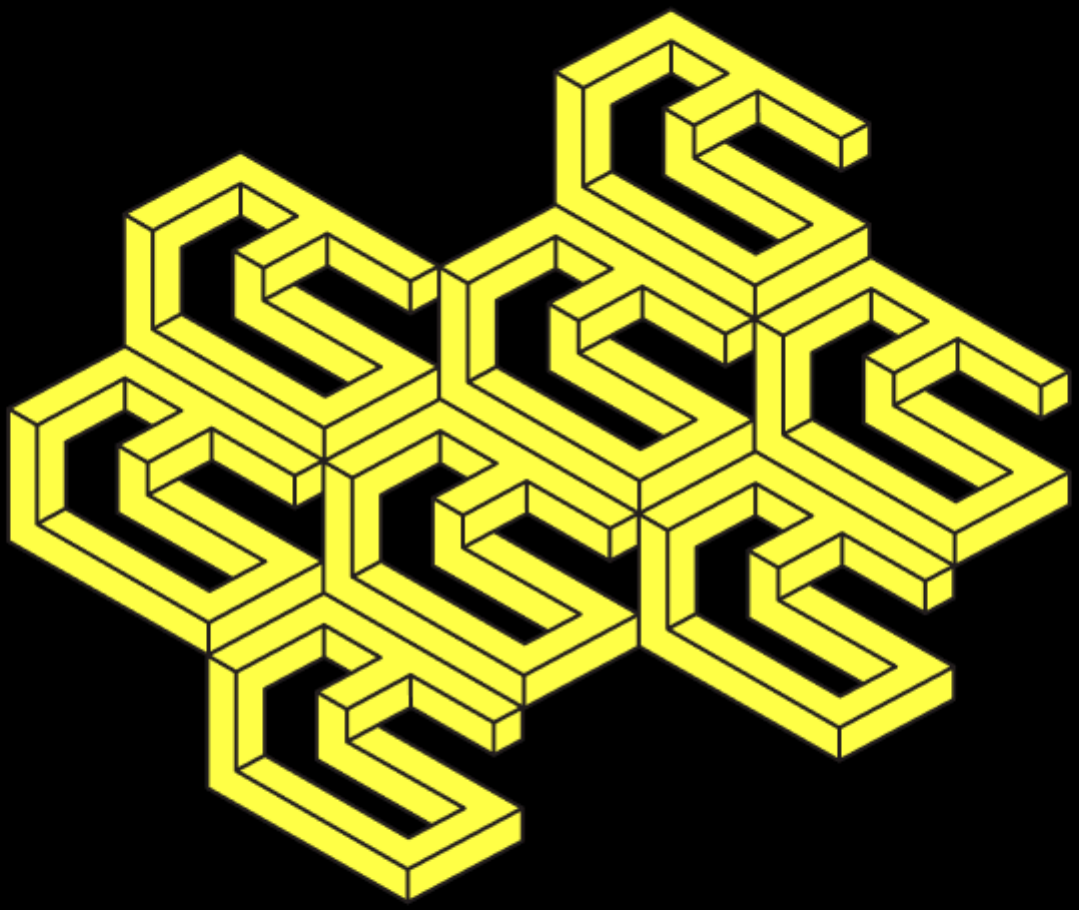

Propy NFT Staking Smart Contract Review | February 2024

by ChainSafe Systems | February 2024

Table of contents

1. Introduction

Date	Auditor(s)
February 2024	Oleksii Matiiasevych, Tanya Bushenyova

Propy requested **ChainSafe Systems** to perform a review of the NFT staking smart contracts. The contracts can be identified by the following git commit hash: `251f78680b92d8bd13977df5db7338de3963f88e`

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

After the initial review, Propy team applied a number of updates which can be identified by the following git commit hash: `7e617249285cf45b33b3be5151f8fc6771e19599`

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 (^0.8.0), that might affect the contracts' logic.

There were **0 critical**, **0 major**, **0 minor**, 14 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/PRONFTStaking.sol

L55 Note Resolved
In the `enter()` function checking `approvedTokenAddresses[_tokenAddress]` could be replaced by checking addresses of `propyNFT` and `ogNFT` and reverting in the `else` case. Other option: `ogNFT` address could be removed and only `approvedTokenAddresses[_tokenAddress]` and `propyNFT` could be checked.

L82 Optimization Resolved
In the `enter()` function the `else if` condition is redundant (the address was already checked as approved and it's not `propyNFT`), `else` would be enough.

L95 Note Resolved
In the `enter()` function the condition where `totalBasePro == 0` but `totalShares != 0` is unreachable. The second part of the `if` condition (`|| totalBasePro == 0`) can be omitted.

L98 Optimization Resolved
In the `enter()` function shares amount could be accumulated and then minted once for the `msg.sender` after the loop.

L99 Optimization Resolved
In the `enter()` function (`if totalShares == 0` condition) `stakerToTotalShares[msg.sender] += amount` can be replaced by `=`.

L101 Optimization Resolved
In the `enter()` function the `what` variable can be replaced by `stakingPowerMinted`.

L104 Optimization Resolved
In the `enter()` function `stakerToTotalShares[msg.sender]` could be updated after the loop with the accumulated value.

L107 Optimization Resolved
In the `enter()` function total amount could be accumulated in the loop, and then `baseProToken.transfer` could be done once after the loop.

L110 Optimization Resolved
In the `enter()` function `stakerToStakedPRO[msg.sender]` could be updated after the loop with the accumulated value.

L111 Optimization Resolved
In the `enter()` function `totalStakedPRO` could be updated after the loop with the accumulated value.

L130 Optimization Resolved
In the `leave()` function the `totalLeftWith` local variable final value is not used so it should be removed.

L143 Optimization Resolved
In the `leave()` function `totalStakedPRO` could be updated after the loop: `tokenAddressToStakedTokenIdToStakedPRO[_tokenAddress][nftId]` could be read once in the loop and accumulated in a local variable, and then `totalStakedPRO` could be updated.

L144 Optimization Resolved
In the `leave()` function `stakerToStakedPRO[msg.sender]` could be updated after the loop: `tokenAddressToStakedTokenIdToStakedPRO[_tokenAddress][nftId]` could be read once in the loop and accumulated in a local variable, and then `stakerToStakedPRO[msg.sender]` could be updated.

L151 Optimization Resolved
In the `leave()` function total amount could be accumulated in the loop, and then `baseProToken.transfer` could be done once after the loop.