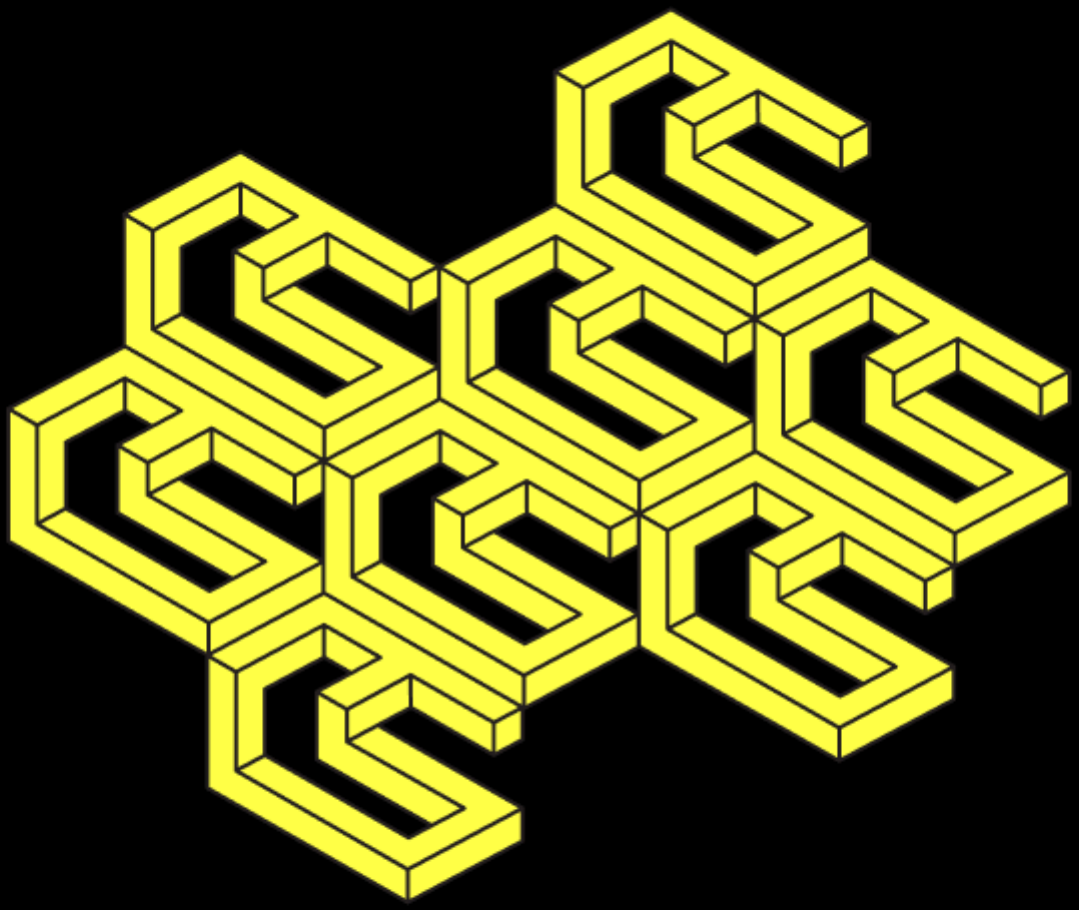

Purpose of Profit Foundation Smart Contracts Review | January 2025

by ChainSafe Systems | January 2025

Table of contents

1. Introduction

Date	Auditor(s)
January 2025	Anderson Lee

Purpose of Profit Foundation requested **ChainSafe Systems** to perform a review of the Purpose smart contracts. The contracts can be identified by the following git commit hash: `5521dbc0544e1bef281afdb9aa4cb3a95ff95991`

There are 3 contracts in scope.

After the initial review, **Purpose of Profit Foundation** team applied a number of updates which can be identified by the following git commit hash: `ccfdd1505e4f8ca1bc5fe60b79fed5b2a39c4d7b`

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

There are **no** known compiler bugs for the specified compiler version (0.8.20), that might affect the contracts' logic.

There were **0 critical**, **0 major**, **1 minor**, 10 informational/optimization issues identified in the initial version of the contracts.

We are looking forward to future engagements with the Purpose of Profit Foundation team.

3. Line-by-line review

src/BuyPurpose.sol

L2 Note Acknowledged

Floating pragma. Consider using the fixed pragma version that was used for development and tests.

L11 Optimization Resolved

The `pfpToken` variable is only set in the constructor. It could be made immutable.

L12 Optimization Acknowledged

The `priceFeed` variable is only set in the constructor. It could be made immutable.

L13 Optimization Resolved

The `TOKEN_PRICE_USD` variable is only set with a hardcoded value during the deployment. It could be made constant.

L45 Minor Acknowledged

```
function getLatestETHPrice() public view returns (uint256) {
    (, int256 answer,,,) = priceFeed.latestRoundData();
    require(answer > 0, "Invalid price from Chainlink");
    return uint256(answer * 10 ** 10); // Convert price to 18 decimals
}
```

The `getLatestETHPrice()` function could return stale results. The Chainlink's `latestRoundData()` function also returns the timestamp which the price was updated but the `getLatestETHPrice()` function doesn't check the timestamp. As a result, the function could return an outdated price. Consider adding a check if the timestamp which the price was updated is within the certain period from the current time.

L54 Note Acknowledged

Deprecated Eth transfer function. Consider using `.call()` function to send ether and check the return value.

src/PurposeRewards.sol

L2 Note Resolved

Floating pragma. Consider using the fixed pragma version that was used for development and tests.

L58 Note Resolved

The `TierFilled` event is not used.

L60 Note Resolved

The `StakerAdded` event is not used.

L230,236 Optimization Resolved

The `tier.stakes[msg.sender]` variable is read twice from storage. Consider caching the value and using it in the function.

L334,340 Optimization Resolved

The `tier.stakes[msg.sender]` is read twice from storage. Consider using the userStake memory variable while updating `rewardDebt`.