Fixed Point Solutions, LLC

Points Tokenization Assessment

2024/04/25
Prepared by: Kurt Barry

1. Scope

A tokenization protocol for points was assessed using 6 hours of effort. The code under review was from the https://github.com/sense-finance/point-tokenization-vault repository at the commit c7a7fd7fa774038b80f7075c3536befb56f0e84e; the files in scope included PToken.sol and PointTokenVault.sol. Dependencies were examined as needed to understand the code but were not explicitly in-scope. The administrator of the system was assumed trustworthy for the security model of this assessment; see note 4.1 for more information.

2. Limitations

No assessment can guarantee the absolute safety or security of a software-based system. Further, a system can become unsafe or insecure over time as it and/or its environment evolves. This assessment aimed to discover as many issues and make as many suggestions for improvement as possible within the specified timeframe. Undiscovered issues, even serious ones, may remain. Issues may also exist in components and dependencies not included in the assessment scope.

3. Findings

Findings and recommendations are listed in this section, grouped into broad categories. It is up to the team behind the code to ultimately decide whether the items listed here qualify as issues that need to be fixed, and whether any suggested changes are worth adopting. When a response from the team regarding a finding is available, it is provided.

Findings are given a severity rating based on their likelihood of causing harm in practice and the potential magnitude of their negative impact. Severity is only a rough guideline as to the risk an issue presents, and all issues should be carefully evaluated.

Severity Level Determination		Impact		
		High	Medium	Low
Likelihood	High	**Critical**	**High**	**Medium**
	Medium	**High**	**Medium**	**Low**
	Low	**Medium**	**Low**	**Low**

Issues that do not present any quantifiable risk (as is common for issues in the Code Quality category) are given a severity of **Informational**.

3.1 Safety and Correctness

Findings that could lead to harmful outcomes or violate the intentions of the system.

SC.1 Users Can Redeem Sufficiently Small Amounts of Rewards Without Burning any PTokens

Severity: Medium

Code Location:
https://github.com/sense-finance/point-tokenization-vault/blob/c7a7fd7fa774038b80f7075c3536befb56f0e84e/contracts/PointTokenVault.sol#L132

Description: When PTokens are burned to claim rewards, truncating division is used:

```
pTokens[pointsId].burn(msg.sender, amountToClaim * 1e18 / rewardsPerPToken);
```

If `amountToClaim` is sufficiently small relative to `rewardsPerPToken`, the burned quantity will be zero and users can redeem rewards while paying only gas costs. Whether this is profitable depends strongly on gas costs and the value of `rewardsPerPToken`, and in many cases it won't be; however, if it is profitable, all the rewards for the relevant `pointsId` can be stolen.

Recommendation: Round up instead of down when calculating the amount of PTokens to burn to eliminate this risk.

Response: Fixed in commit 850c5626ab615eefcfc4577047a620afd343473f.

FPS: Fix verified.

SC.2 Problems if the Points Value Assigned to the Protocol Contract Decreases

Severity: Low

Code Location: N/A (mechanism-level)

Description: If for any reason the points quantity assigned to the protocol contract decreases below the amount of PTokens minted, there is no way to gracefully compensate for this. PTokens will begin to trade at a discount based on the shortfall between PToken supply and actual points assigned, resulting in losses to the users holding them when the decrease happens. Since points are typically implemented in a centralized manner, there is no guarantee that this will not happen, and the protocol could even be targeted intentionally by projects that don't appreciate the financialization of points.

Recommendation: While this is likely a risk that must be accepted to a certain extent, it would be prudent for operators of the system to seek assurances from points-assigning entities prior to enabling new PTokens.

Response: Risk understood, pursuing mitigations where possible.

SC.3 Various Non-Standard Token Behaviors Are Unsupported

Severity: Low

Code Location: throughout PointTokenVault.sol

Description: Non-standard token behaviors like fee-on-transfer and rebasing are incompatible with the PointTokenVault contract as written, as it assumes deposited token balances do not change over time and equal the deposited amount post-transfer. Further, the deposit function is permissionless so there is no way to prevent users from accidentally depositing incompatible tokens and suffering a loss. Reward tokens with these behaviors are also problematic.

Recommendation: While no action is strictly necessary, considering warning users in the UI and recommending wrapped versions of any incompatible tokens. Incompatible reward tokens must also be wrapped

Response: Acknowledged, documented in code comments and will warn users in the UI.

3.2 Usability and Incentives

Findings that could lead to suboptimal user experience, hinder integrations, or lead to undesirable behavioral outcomes.

U.1 Loss of PToken Liquidity After Redemptions Are Enabled May Harm Financial Use-Cases

Severity: Low

Code Location: N/A (mechanism-level)

Description: The loss of PToken liquidity due to PTokens being burned to claim reward tokens may harm certain financial use-cases. Concretely, users who borrow and then sell (i.e. short) PTokens may be unable to close their positions if they do not buy back PTokens quickly enough. This could cause losses for users with otherwise profitable positions, result in pricing distortions, and discourage users from taking full advantage of the use cases PTokens are in principle intended to enable.

Recommendation: Consider allowing users to convert reward tokens to PTokens once a `rewardsPerPToken` value has been established. Note that this will not work if multiple reward tokens are distributed per point.

Response: Solution implemented in aed4f4addf8f674584c41f30b38720d286becbc8.

FPS: Fix verified.

U.2 Tokens With Redemption Rights May Inconvenience Users and Result in Market Distortions

Severity: Low

Code Location: N/A (mechanism-level)

Description: To support reward tokens with vesting schedules, the protocol includes a notion of “redemption rights” that are assigned to users as the reward tokens corresponding to the points earned by the tokens they deposited vest. This impairs the fungibility of PTokens to some extent, and requires specific actors to take action to unlock reward tokens. It risks either that users will be reluctant to sell or lend their claimed PTokens, as they fear being squeezed when trying to buy back to redeem, or that others will be reluctant to buy PTokens as they will not be able to redeem them without a rights-holding user being willing to take action.

Recommendation: Consider alternative ways to handle vesting or additional features. Allowing PTokens to be redeemed by anyone up to the currently vested amount is one option, although it can create a potentially value-destructive “race to redeem”. Another option would be to allow users to delegate their redemption rights to another address. This would be the minimal functionality needed to allow building a trustless market for redemption rights on top of the PointTokenVault, although allowing PTokens and redemption rights to be separately tradeable may be a complex experience for users. An option that would solve this but involve more extensive changes would be issuing NFTs instead of fungible PTokens when vesting or other fungibility-breaking features are present.

Response: Acknowledged, will consider changing in the future based on how the initial launch goes.

3.3 Gas Optimizations

Findings that could reduce the gas costs of interacting with the protocol, potentially on an amortized or averaged basis.

N/A. No significant gas optimizations were found, and minor efficiency improvements were not a focus of this assessment.

3.4 Code Quality

CQ.1 Link to Mutable External Code

Severity: **Informational**

Code Location:
https://github.com/sense-finance/point-tokenization-vault/blob/c7a7fd7fa774038b80f7075c3536befb56f0e84e/contracts/PointTokenVault.sol#L97

Description: The link in this comment points to the main branch of an external repository; over time, the external code may change and no longer match what was originally intended, causing confusion for readers of the code.

Recommendation: Consider linking to a specific commit hash instead.

Response: Fixed in commit aed4f4addf8f674584c41f30b38720d286becbc8.

FPS: Fix verified.

4. Notes

This section contains general considerations for interacting with or maintaining the system and various conclusions reached or discoveries made during the course of the assessment. Whereas findings generally represent things for the team to consider changing, notes are more informational and may be helpful to those who intend to interact with the system.

4.1 High Degree of Centralization

The protocol relies on a trusted admin to set reward rates, claim rewards, perform upgrades, and manage authorizations. The security model for this assessment assumed this admin was trustworthy. Users should be aware the admin has the ability to make arbitrary changes to the protocol, including the power to steal all deposited tokens, steal all rewards, and upgrade the contract to be able to burn PTokens from any user. The team has indicated they plan to pursue more censorship-resistant administrative mechanisms in the future.