SOLIDIFIED

Audit Report for The Sandbox Starter Pack Sale - October 22, 2020

Summary

Audit Report prepared by Solidified covering The Sandbox reward pool smart contracts (and their associated components).

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the code below. The debrief took place on October 20th, 2020.

Replies and fixes were received on October 21st, 2020. The final results are presented here.

Audited Files

The following contracts were covered during the audit:

- src/LiquidityMining/LandWeightedSANDRewardPool.sol

Supplied in the private repository:
https://github.com/thesandboxgame/sandbox-private-contracts

Notes

The audit was based on commit `1768142603a948bd7492cc87a66e19160489043f`

UPDATE: Fixes were received in commit `c096223e27a736a103324e299c443699843a5a71`

Intended Behavior

The reward pool implements a reward mechanism for stakers. It builds on the unipool contract by Synthetix (https://github.com/Synthetixio/Unipool/blob/master/contracts/Unipool.sol) and provides a mechanism to distribute rewards based on the amount of ERC-721 Land a staker possess.

SOLIDIFIED

Audit Report for The Sandbox Starter Pack Sale - October 22, 2020

Executive Summary

Solidified found that the Sandbox contracts contain 3 major issues, 1 minor issue, in addition to 1 informational note.

We recommend all issues are amended, while the notes are up to the team's discretion, as it refers to best practices.

Issues found:

Issue #	Description	Severity	Status
1	Total contribution is not reduced when stakes are withdrawn	Major	Resolved
2	Land owned may not be accurate during reward calculation	Major	Acknowledged
3	Potential reentrancy from token contract	Major	Resolved
4	There is no check that ensures the contract received the amount passed in `notifyRewardAmount`	Minor	Acknowledged
5	Reward calculation performed twice	Note	Resolved

Critical Issues

No critical issues have been found.

Major Issues

1. Total contribution is not reduced when stakes are withdrawn

The `withdraw()` function reduces the contribution of `msg.sender`, but fails to update the `_totalContributions` variable. This leads to incorrect pool shares and, therefore, incorrect reward distributions.

Recommendation
Subtract the withdrawn amount from `_totalContributions`.

2. Land owned may not be accurate during reward calculation

`LandWeighterSANDRewardPool.sol`: Contributions are calculated using the number of land NFT at the time of staking. Since the number of land is not locked after staking, this value can change anytime. Apart from not reflecting changes and introducing inconsistencies, this could be exploited by transferring land between addresses to increase individual contributions.
In the case of land being available on an open market, the impact could increase significantly with flashloans.

Recommendation
Consider changing the reward mechanism to require LAND to be staked, as well as SAND.

Team Response
“This is as expected and since the user loses some of the contribution share by loaning its LANDs to others (as they can use it to increase their share), we do not think this is actually economically sensical to do it. Furthermore, you still need to stake SAND
For these reasons, we think the current behavior is fine
It also has the benefit to allow Land owners to retain the ownership of their Land during the reward period.”

3. Potential reentrency from token contract

`LandWeighterSANDRewardPool.sol`: The stake() and withdraw() function call their equivalents in the LPTokenWrapper contract, which in turn calls the external token. If the token is untrusted or allows untrusted code to be executed (as in some implementations), this may lead to a potentially exploitable reentrancy.

Recommendation
It is recommended to move the `super.stake()` and `super.withdraw()` interactions towards the end of the function.

Minor Issues

4. There is no check that ensures the contract received the amount passed in `` `notifyRewardAmount` ``

It is possible for this function to be called with a different amount than what was sent to the contract, possibly making rewards insufficient for all stakers.

Recommendation
Add a mechanism to verify that the pool contract holds the amount passed in this function

Team Response
“We leave as is, as this allows more flexibility in setting this up.”

Notes

5. Reward calculation performed twice

The function `getReward()` calls the `earned()` function twice, once through the `updateReward()` modifier, and then again immediately, introducing unnecessary logic to be executed.

Recommendation

Instead of calling `earned()` again in the function body, the user's rewards can be accessed directly through `rewards[msg.sender]`, since this has been updated by the modifier already.

Audit Report for The Sandbox Starter Pack Sale - October 22, 2020

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of TSB GAMING LTD or its products. This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.