By
Quit

Protocol Audit
Issue date
12/21/2022

Overview

ParaSpace recently released auto-compounding for ApeCoin Staking Pool 0, also known as the ApeCoin only pool. ApeCoin holders can deposit through ParaSpace's auto-compounder, which in turn pools everybody's tokens under a single "owner", allowing for what is effectively a batch compounding mechanism that is triggerable by anybody at any time (given the amount to compound is at least 100 ApeCoin).

Thoughts

Batch compounding is a hot topic as of late. The implementations are fairly simple, the consequence of exploit is especially high. In this review I'll be attempting to identify any potential exploits involving:

- Unpermissioned withdrawal of balances by exploiter or contract owner
- Manipulation of pool shares to the benefit an exploiter
- Manipulation of pool shares to adversely affect a given user
- Unwanted behavior for when `_harvest` or `_compound` are called
- Any potential cause for loss of ApeCoin

Findings

`rescueERC20`: Prefer to place require statements before state updates

Severity: QA/QOL

It is generally best practice to place any require statements at the top of your functions. This pattern protects against re-entrancy, but also saves gas in case of revert. This particular function is not vulnerable to re-entrancy, but it is still recommended to follow the pattern and adjust the require accordingly (typically with a new pointer stored in memory).

Use of Require Instead of Custom Errors

Severity: Gas Optimization

This was touched on in the initial audit, but reiterating here: Strings are very expensive in solidity, which makes require(condition, error_string) less than ideal. Instead, opt to use custom errors which cut down on deploy and revert costs.

SafeMath is effectively obsolete in Solidity 0.8+

Severity: QA/QOL

Starting with solidity 0.8, over and underflow checks are included by default. SafeMath may have slightly cheaper checks in this area, but it is recommended to omit it outside of a few edge cases (which are not present here).

Use Pre-Increment Operator And Unchecked Blocks Where Possible
Severity: Gas Optimization

Over/underflow is not a realistic concern when looping in `PoolApeStaking.claimApeAndCompound`. Using pre-increment operators and unchecked blocks here can save a small amount of gas.

Upgrading to Solidity 0.8.18 may give slight gas improvements
Severity: QA/QOL

Comment accuracy
Severity: QA/QOL
`AutoCompoundApe.sol:16 - Ape coin` should read `ApeCoin`
`CApe.sol:30 - Paraspece` should read `ParaSpace`
`CApe.sol:241 - Ether amount` should read `ApeCoin amount`

Conclusion

The auto-compounder doesn't have any apparent vulnerabilities. Users can be reasonably assured that in its current state, the contract is safe to operate through for more efficient ApeCoin only staking. Bear in mind that this contract is still upgradeable, and logic changes can introduce either new bugs, or malicious code.