Ēnosys Stake Helper

Complementary Smart Contract Audit

June 28, 2023

Common Prefix

Overview

Introduction

Common Prefix was commissioned to perform a security audit on Ēnosys's Stake Helper smart contracts, at commit hash f93cb958647bce1e4c6b7ffd7f10f6fe1f834162. We have previously audited the Stake Helper protocol[1]. The scope of the current complementary audit is restricted to the newly added features related to protocol fees. The files inspected are the following:

`PoolHelperBase.sol`

`PoolStakeHelper.sol`

`PoolWithdrawHelper.sol`

Disclaimer

Note that this audit does not give any warranties on the bug-free status of the given smart contracts, i.e. the evaluation result does not guarantee the nonexistence of any further findings of security issues. This audit report is intended to be used for discussion purposes only. Functional correctness should not rely on human inspection but be verified through thorough testing. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of the project.

Findings Severity Breakdown

The findings are classified under the following severity categories according to the impact and the likelihood of an attack.

Level	**Description**
Critical	Logical errors or implementation bugs that are easily exploited and may lead to any kind of loss of funds

[1] The report of the previous audit can be found here: FLRFinance StakeHelper

High	Logical errors or implementation bugs that are likely to be exploited and may have disadvantageous economic impact or contract failure
Medium	Issues that may break the intended contract logic or lead to DoS attacks
Low	Issues harder to exploit (exploitable with low probability), issues that lead to poor contract performance, clumsy logic or seriously error-prone implementation
Informational	Advisory comments and recommendations that could help make the codebase clearer, more readable and easier to maintain

Findings

Critical

No critical issues found.

High

No high issues found.

Medium

MEDIUM-1	Erroneous amount computation
Contract(s)	*PoolStakeHelper.sol*
Status	**Resolved**

Description

In `PoolStakeHelper::getAmountsLpFarmWithExactMultiTokens` it is checked if the token denoted as `token0` by the user is actually the `token0` of the pair and, if it is not, `token0` and `token1` and the corresponding amounts change order. But in the code what it is actually done is the following:

```
if (token0 != token0Pair) { //incorrect order, we should set the correct one
         token0 = token0Pair;
         token1 = token1Pair;
         uint correctToke1Amount = amount0;
         amount0 = amount1;
         amount0 = correctToke1Amount;
     }
```

The result of this code is that the amounts do not change order.

Recommendation

We suggest replacing `amount0` with `amount1` in the last line of the code snippet above.

Alleviation

The team fixed the issue at commit hash edca42997104fbb3ecda7b94210104c0e576294d.

Low

LOW-1	Problem with staking if the user deposits WNAT and NAT tokens
Contract(s)	*`PoolStakeHelper.sol`*
Status	**Resolved**

Description

In `PoolStakeHelper::stakeLpFarmWithExactTokenAndFlr` if the exact token is WNAT, there will be a problem, because this function accepts an exact token and NAT tokens which transforms to WNAT, therefore if the exact token it WNAT, both `token0` and `token1` of the pair will be the WNAT token.

Recommendation

We suggest adding a `require(token!=address(WNAT))` to avoid this case.

Alleviation

The team fixed the issue at commit hash 6a31ed45f5c6b930fde65d7a23aa8a789ca6bdd9.

Informational/Suggestions

INFO-1	Typos
Contract(s)	*`PoolHelperBase.sol`*, *`PoolStakeHelper.sol`*
Status	**Resolved**

Description

There are a few typos in the names of variables and functions.

In `PoolHelperBase.sol:`

- `indifidualPoolFees`
- `_estimeteSwap`

In `PoolStakeHelper.sol`:

- `_estimeteProvidingLp()`
- In `getAmountsLpFarmWithExactMultiTokens`: `correctToke1Amount`

Recommendation

We suggest fixing these typos to increase readability.

Alleviation

The team fixed the typos at commit hash 3a8ae85832fb129afe76ed1f4baca93b90234570.

INFO-2	Unnecessary type casting
Contract(s)	*`PoolHelperBase.sol`*
Status	**Resolved**

Description

In PoolHelperBase::_updateKakeibo() there is the following line

```
  require(_kakeibo != address(kakeibo), "FarmingPoolHelper: The same kakeibo");
```

But the variable `kakeibo` is of type address, therefore the type casting is redundant.

Recommendation

We suggest removing the type casting.

Alleviation

The team fixed the issue at commit hash 5a6b41e4a0c8affe64490ab55708a1a4b19bf4bb.

INFO-3	Convert WNAT to Nat
Contract(s)	*PoolStakeHelper.sol*
Status	**Resolved**

Description

In `PoolStakeHelper::_stakeLpFarmWithExactTokenAndSwap()` the function _stakeToLpFarm is being called with the argument convertToNativeToken=false , but if `IsInAssetNativeToken==true` maybe the unused WNAT should be converted to NAT, as it is done in all the other cases

Recommendation

We suggest fixing this issue making the code treating all the cases uniformly.

Alleviation

The team fixed the issue at commit hash f383577f4b4a842f4afbc12c8156e360aa612684.

INFO-4	Avoid "magic" constants
Contract(s)	*PoolStakeHelper.sol*

Status	**Resolved**

Description

In `PoolStakeHelper:: _estimeteProvidingLp` the number 10**3 appears corresponding to the minimum liquidity amount of a pool.

Recommendation

We suggest replacing this hardcoded number with a constant to increase readability.

Alleviation

The team fixed the issue at commit hash 304154d224a7822f5da933a07b4bedda98bb3199.

INFO-5	Allow different minimum amounts
Contract(s)	*`PoolWithdrawHelper.sol`*
Status	**Resolved**

Description

The function `withdrawLpToken()` has only one `minAmounToReceive` argument although there are two swap paths.

Recommendation

We suggest allowing two minimum amounts one for each path to increase flexibility.

Alleviation

The team fixed the issue at commit hash 9dd72dbfe9c785469d1c6739b3bc3ae653f6a5e6.

INFO-6	Missing case in `_swapToken()` could cause problems in future versions
Contract(s)	*`PoolStakeHelper.sol, PoolWithdrawHelper.sol`*
Status	**Acknowledged**

Description

The router reverts, if you provide a path with length 1, therefore `PoolStakeHelper,PoolWithdrawHelper::_swapTokens` cannot work if the `path.lenth` is 1. Although it is not a problem in the current implementation because the paths passed as arguments to this have always a length greater than one, it could be a problem if the code is updated in the future adding extra functionality.

Recommendation

We suggest fixing this edge case to avoid future problems.

About Common Prefix

Common Prefix is a blockchain research, development, and consulting company consisting of a small number of scientists and engineers specializing in many aspects of blockchain science. We work with industry partners who are looking to advance the state-of-the-art in our field to help them analyze and design simple but rigorous protocols from first principles, with provable security in mind.

Our consulting and audits pertain to theoretical cryptographic protocol analyses as well as the pragmatic auditing of implementations in both core consensus technologies and application layer smart contracts.