



SMART CONTRACT AUDIT

Certification

FARM CONTRACT



SMART CONTRACT AUDITS

Feb 14th, 2024 / v.0.2

Audited source code version:

b427228cd3d9e9bf4f7d567eb0ca4fc92a726d88

Structure and Organization of the Document

Some sections are more important than others. The most critical areas are at the top, and the less critical sections are at the bottom. The issues in these sections have been fixed or addressed and will show by the "Resolved" or "Unresolved" tags. Each case is written so you can understand how serious it is, with an explanation of whether it is a risk of exploitation or unexpected behavior.

CRITICAL

These issues can have a dangerous effect on the ability of the contract to work correctly.

HIGH

These issues significantly affect the ability of the contract to work correctly.

MEDIUM

These issues affect the ability of the contract to operate correctly but do not hinder its behavior.

LOW

These issues have a minimal impact on the contract's ability to operate.

INFORMATIONAL

These issues do not impact the contract's ability to operate.

Issues

1. Unchecked input

Fixed / **CRITICAL**

Description: The endpoint '**changeAnnualReward**' does not check if the pool actually has a second reward token configured upon receiving a value for '**year_second_reward_amount**'. This can lead to problems because this misconfiguration can cause the contract to try to send rewards (in second token) to users and the send will fail, thus resulting in users not being able to claim their rewards.

! Possible fix to research

When trying to configure a value for the second token reward amount, check that a second token is configured in the particular pool. Also, make sure the value, when supplied, is greater than zero (as this enforcement is set for the first reward token).

! Response

Fixed.

! Status

Accepted & Closed

2. Out of gas / read operations

Fixed / HIGH

Description: When calculating the reward for a user, the smart contract does an iteration over all the **'RewardInfo'** structs. A new **'RewardInfo'** struct is created each time the pool owner changes the APR. Assuming that there are few changes in the APR, this should not be a problem, but if the vector grows, user funds are at risk of being locked in the contract forever.

! Possible fix to research

The problem appears when the pool owner changes the APR very frequently. Add guards such that there's a cooldown between the last change in APR so the contract will try to diminish the amounts of **'RewardInfo'** created.

! Response

An only-owner function was introduced that can be used to clear old entries of the **'RewardInfo'** vector.

! Status

Accepted & Closed

3. Unchecked input

Fixed / LOW

Description: When creating a new pool, two tokens are required, which are checked using **'is_valid_esdt_identifier'** and **'is_valid'** and a third one is optional, although if it is provided, its validity is not checked.

! Possible fix to research

Check the validity of the third token when it is supplied using **'is_valid'** function.

! Response

Fixed.

! Status

Accepted & Closed

4. Fees Leftover

Fixed / **LOW**

Description: The contract splits the fee retained when creating a new pool into two. One half to the **'treasury_address'** and the other to **'burner_address'**. However if the fee amount is an odd number, a rounding error of **'1'** will remain in the contract.

! Possible fix to research

Fix: Send `'amount / 2'` to one address and send `'amount - amount / 2'` to the other in order to avoid encountering leftovers.

! Response

Fixed.

! Status

Accepted & Closed

5. Unused decimals

Not addressed / **LOW**

Description: When a pool is created, **'reward_token_decimal'** and **'second_reward_token_decimal'** are required, however they are not being used in the contract.

! Possible fix to research

Either use the values where it was intended doing the development or remove them.

! Response

Not addressed.

! Status

Open issue.

6. Commented code

Not Addressed / **INFORMATIONAL**

! Status

Not Addressed.

7. Empty events module

Not Addressed / **INFORMATIONAL**

! Status

Not Addressed.

8. Lack of tests

Not Addressed / **INFORMATIONAL**

! Status

Not Addressed.

9. Lack of interaction scripts

Not Addressed / **INFORMATIONAL**

! Status

Not Addressed.

10. Lack of documentation

Not Addressed / **INFORMATIONAL**

! Status

Not Addressed.

Verification Conditions

1 Owner functions are marked using either 'only_owner' macro attribute.

```
#[only_owner]
#[endpoint(setCreationCost)]
fn set_pool_creation_cost(
```

2 Pool admin functions are guarded correctly.

```
self.assert_pool_owner(pool_id);
```

3 Valid payments are checked on input.

```
require!(
    stake_token_id == self.pool_stake_token_id(pool_id).get(),
    "Invalid lp token id"
);
```

4 Actions are taken when the contract and the pool are not paused (and only on valid pools).

```
self.assert_valid_pool_id(pool_id);
self.assert_unpaused();
self.assert_pool_unpaused(pool_id);
```

Suggestions (Optional)

1. Update to latest framework version (best practice when deploying a new contract).
2. Write tests (Rust Testing Framework is recommended, documentation (at least a readme) and interaction scripts).
3. Format the code using '**cargo fmt**' and solve the warnings (check them using '**cargo clippy**').
4. Construct the '**Pool**' structure and store it inside a single storage instead of splitting pieces of information between multiple storages.

Test results

There are no tests.

Audited source code version

633c2a14b4680a95b4c7d238cd24929aff91d7dc

Second Audited source code version

b427228cd3d9e9bf4f7d567eb0ca4fc92a726d88