SMART CONTRACT AUDIT

Certification

ESDT TOKENS LIQUID STAKING CONTRACT

SMART CONTRACT AUDITS

Apr 1st, 2024 / v.0.2

Audited source code version:

f0238483e83a3039779da668dbb74778c090dd38

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. Loss of funds

Fixed / CRITICAL

Description: Anyone can drain all the tokens inside the contract, the contract allows for external handling without verifications.

! Possible fix to research

```
Remove the '#[endpoint]' on the 'unwrap' function.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

2. Loss of funds

Fixed / CRITICAL

Description: The contract tries to generate rewards even after the end date, which will result in error (negative value after subtraction on BigUint rewards_reserve) that will prevent users from unstaking their tokens.

! Possible fix to research

```
Change the "&&" to "||" in "if current_timestamp > deadline && current_
timestamp < start_rewards".
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

3. Unfair reward distribution

Fixed / MEDIUM

Description: When adding or removing tokens via top up or regress, the stakes must be rewarded with the rate that was available before the change, in order to preserve the terms and meet their expectation. Updating the rate and then doing the recalculation for the future might be unfair for some users.

! Possible fix to research

```
First thing to do in both topup and regress is to generate the aggregated
rewards. After this is done, the new rate can be calculated and
installed.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

4. Unchangeable tax cuts

Fixed / MEDIUM

Description: Once set, the tax cut percentages cannot be changed. The function does clear old entries and cannot add new ones since the sum of percentages together with the new entries will have to remain 100.

! Possible fix to research

```
Either move the setting of percentages in the init function, in case
it is meant to be used only once, or clear the map before setting new
entries again, in case it is meant to be reused in the future.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

5. Contract state not respected

Fixed / MEDIUM

Description: Some of the public endpoints (for example topup and regress) are callable even if the contract state is inactive.

! Possible fix to research

```
Make public endpoints uncallable while inactive.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

6. Topup and Regress might not generate rewards

Fixed / LOW

Description: The two endpoints do not generate aggregated rewards while the pools are paused.

! Possible fix to research

```
Remove the check for pool state from 'calculate_rewards_since_last_
allocation' function.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

Verification Conditions

1 Admin functions are marked using

```
#[only_owner]
```

2 User actions are guarded by active state checks on the contract and the pool.

```
require!(
    self.contract_state().get() == State::Active,
    ERR_CONTRACT_IS_INACTIVE
);

require!(
    self.is_state_active(storage_cache.pool_state),
    ERR_POOL_IS_INACTIVE
);
```

3 Valid tokens are accepted on adding liquidity

```
require!(
    payment.token_identifier
        == self
            .stake_pool_rewards_token_identifier(&stake_pool_id)
            .get(),
    ERR_INVALID_TOKEN
);
```

4 Pool management functions can be accessed only by the owner

```
require!(
  self.is_stake_pool_owner(&stake_pool_id, &caller),
  ERR_INVALID_STAKE_POOL_OWNER
);
```

Suggestions (Optional)

1. Fix clippy warnings (for example there are plenty regarding double referencing).

2. Consider either moving all error messages into errors.rs or have them all inline for improving the code style.

Test results

```
running 11 tests
test test_deploy_and_pause ... ok
test percentage_equal_max_percentage ... ok
test top_up_with_esdt_and_deadline ... ok
test percentage_under_max_percentage - should panic ... ok
test percentage_over_max_percentage - should panic ... ok
test regress_with_esdt_payment_and_new_deadline ... ok
test top_up_with_esdt ... ok
test regress_with_esdt_payment ... ok
test create_stake_pool ... ok
test stake ... ok
test unstake_and_unbound ... ok

test result: ok. 11 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.04s
```

Audited source code version
3c75e232645881b4da2c809cd9a665b9e42e727e

Second audited source code version
f0238483e83a3039779da668dbb74778c090dd38