SMART CONTRACT AUDIT

Certification

STAKING CONTRACT

SMART CONTRACT AUDITS

March 22nd, 2023 / v.0.2

Audited source code version:

e4b49a864bd4ee3872cb7ea0ace2c2a6ef62222a

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 / HIGH

Description: **'refillRewardFund'** function takes as input **'staking_token_identifier'** instead of **'reward_token_identifier'**. Imagine the scenario where the **'staking_token'** differs from the **'reward_token'**. In this scenario, the tokens deposited by the owner would be lost forever. Also, the users would not be able to receive their rewards for staking their tokens.

! Possible fix to research

```
Make the function accept ‘reward_token’ instead of ‘staking_token’.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

2. Loss of funds

Fixed / HIGH

Description: The penalty fees for unstaking too early are forever locked in the contract. Namely the **'recovered_penalty_amount'** storage is just increased in **'unstake_common'** and never used afterwards.

! Possible fix to research

```
Make an ‘only_owner’ function that does something with those tokens (eg.
withdraws them).
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

3. Misscheck of configuration

Fixed / HIGH

Description: **'require_staking_is_configured'** function does not check if the **'reward_per_block'** is different from zero, as expected in other functions such as **'setRewardsPerBlock'**.

! **Possible fix to research**

```
Add a check in ‘require_staking_is_configured’ and make sure ‘reward_per_block’
is greater than zero.
```

! **Response**

```
Fixed.
```

! **Status**

```
Accepted & Closed.
```

4. Misscheck of configuration

Fixed / MEDIUM

Description: **'calculate_reward_per_token'** works if the **'total_supply'** is zero, even if the contract is not configured.

! **Possible fix to research**

```
Move the call to ‘require_staking_is_configured’ function at the top of the
function.
```

! **Response**

```
Fixed.
```

! **Status**

```
Accepted & Closed.
```

5. Rewards are not distributed

Fixed / MEDIUM

Description: **'claim_rewards_for_user'** and **'stake_rewards'** check if the **'reward_fund'** is strictly greater than the reward one user might be eligible. If the two are equal, the rewards are not given to the user.

! Possible fix to research

Change the comparison operator from **'>'** to **'>='** between the **'reward_fund'** and **'reward_amount'**

! Response

Fixed.

! Status

Accepted & Closed.

6. Unchecked precision

Fixed / MEDIUM

Description: The logic misbehaves if the **'division_safety_constant'** has low values. This can happen if the **'precision'** argument has low values.

! Possible fix to research

In the **'init'** function, add a check for a minimal value of **'precision'**. Right now, even the zero value is accepted (which makes **'division_safety_constant'** equal to 1 and hence redundant).

! Response

Fixed.

! Status

Accepted & Closed.

7. Unstaking with maximum penalty fails

Fixed / MEDIUM

Description: **'unstake_common'** misses a check for zero value when sending back the staked value after penalty, namely the **'amount_for_user'**. The maximum allowed penalty fee is 100% of the staking position. In case this is configured and a user still wants to unstake for some reason, he wouldn't be able to.

! Possible fix to research

```
Add a check for 'amount_for_user'. Do the transfer only if the value is greater
than zero.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

8. Unchangeable precision

Fixed / LOW

Description: Once set, the **'division_safety_constant'** cannot be changed afterwards. Even though it is of low probability, it may be useful to change it (may need in case **'reward_per_token'** grows much).

! Possible fix to research

```
Make an 'only_owner' function that can set the value.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed (Boundaries have been set on initialization).
```

9. Gas optimization

Fixed / LOW

Description: **'stake_rewards'** and **'claim_rewards_for_user'** functions have reset the **'reward_per_user'** storage for a user. Both use **'set(BigUint::zero())'** which means creating a new instance of **'BigUint'** and attempting to write it to the storage.

! Possible fix to research

Use **'clear()'**. It does the same thing without the need of a new instance of **'BigUint'**.

! Response

Fixed.

! Status

Accepted & Closed.

10. Gas optimization

Fixed / LOW

Description: There are a few flows where the current block is read multiple times. Additionally, there are a few flows where user rewards are written to the storage and then read afterwards. These operations cost gas, and just by refactoring the code by passing those values from function to function using parameters or a more generic cache, execution time can be improved and gas cost can be reduced.

! Possible fix to research

Refactor functions to remove redundant multiple reads and write-then-reads.

! Response

Fixed.

! Status

Accepted & Closed.

11. Gas optimization

Fixed / LOW

Description: Avoid instantiation of **'BigUint::zero()'** when doing comparisons.

! Possible fix to research

```
When doing comparisons, put the value of type 'BigUint' in the left side. This
way the compiler would allow the '0u64' to be on the right side, without the
need of a new 'BigUint' instance
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

12. Storage layout differs in format

Fixed / INFORMATIONAL

Description: **'stakedAddresses'** and **'stakingPosition'** storage mappers have camel case format and the rest have snake case format.

! Possible fix to research

```
In order to be consistent, make all of them either snake case or camel case.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

13. Remove hardcoded values

Fixed / INFORMATIONAL

Description: The **'1e18'** value is used in **'calculate_reward_per_token'** and **'calculate_rewards'** and it might not be intuitive what it represents.

! Possible fix to research

```
Remove the hardcoded value by using a constant. It gives it a name and more
meaning. Also, changing it later will persist everywhere in the code.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

Verification Conditions

1 'stake_through_proxy' and 'unstake_through_proxy' require whitelisting

```
let caller = self.blockchain().get_caller();
self.known_staking_proxy_contracts()
    .require_whitelisted(&caller);
```

2 'stake_common' does the appropriate checks on token id and amount.

3 Contract configuration endpoints have only_owner annotation.

4 'penalty_percentage' boundaries are checked when attempting to set them.

```
require!(
    PENALTY_MAX_PERCENTAGE >= penalty_percentage,
    “Percentage must be between 0 and 10000”
);
```

Suggestions (Optional)

1. Rename **'unwhitelistStakingProxyContract'** to **'removeWhitelistStakingProxyContract'** or similar.

```
Response: Not addressed.

Status: Accepted & Closed (Informational).
```

2. When adding or subtracting **'BigUints'** from each other, try using references rather than the values directly, because they move the value and usually require **'clone()'** calls.

```
Response: Fixed.

Status: Accepted & Closed.
```

3. Update the MultiversX Rust Framework version.

```
Response: Fixed.

Status: Accepted & Closed.
```

4. Write tests using Rust Testing Framework instead of Mandos json files.

```
Response: Not addressed.

Status: Accepted & Closed (Informational).
```

5. Add tests for the scenario where the reward token is different from the staking token.

```
Response: Not addressed.

Status: Closed.
```

6. Re-make the test **'configure-refill-reward-fund-failed-invalid-token-payment.scen'**.

```
Response: Not addressed.

Status: Closed.
```

7. Refactor the **'unstake_common'** function. It was 72 lines and lots of **'if'** statements.

```
Response: Fixed.

Status: Accepted & Closed.
```

8. Add a way of stopping and starting to generate the rewards. Might be useful in emergency cases.

```
Response: Fixed.

Status: Accepted & Closed.
```

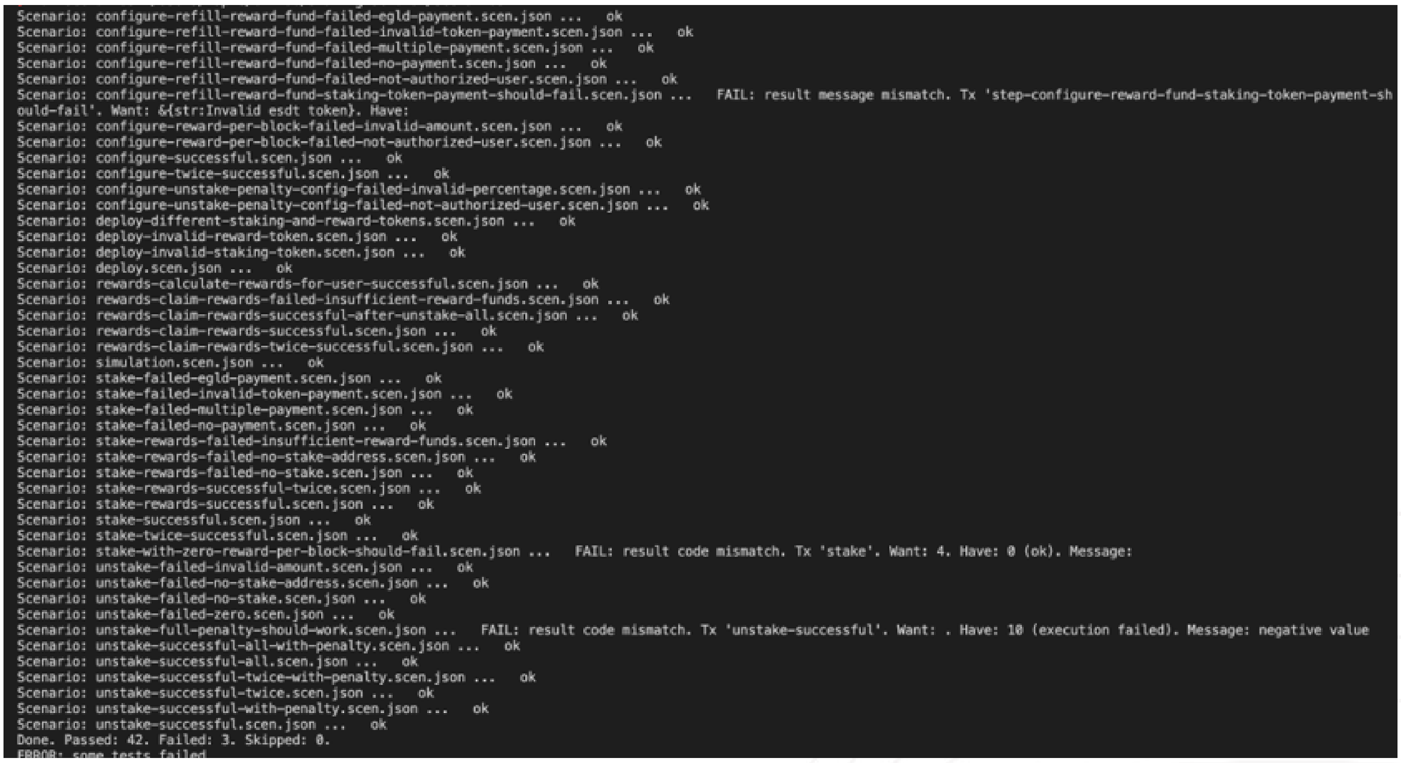
9. Do not use intermediary storage mapper objects. When in need to call **'is_empty'** or **'get()'** or **'set()'**, invoke the mapper inline.

```
Response: Fixed.

Status: Accepted & Closed.
```

10. In the **'unwhitelist_staking_proxy_contract'** function, use the **'require_whitelisted'** method of **'WhitelistMapper'**.

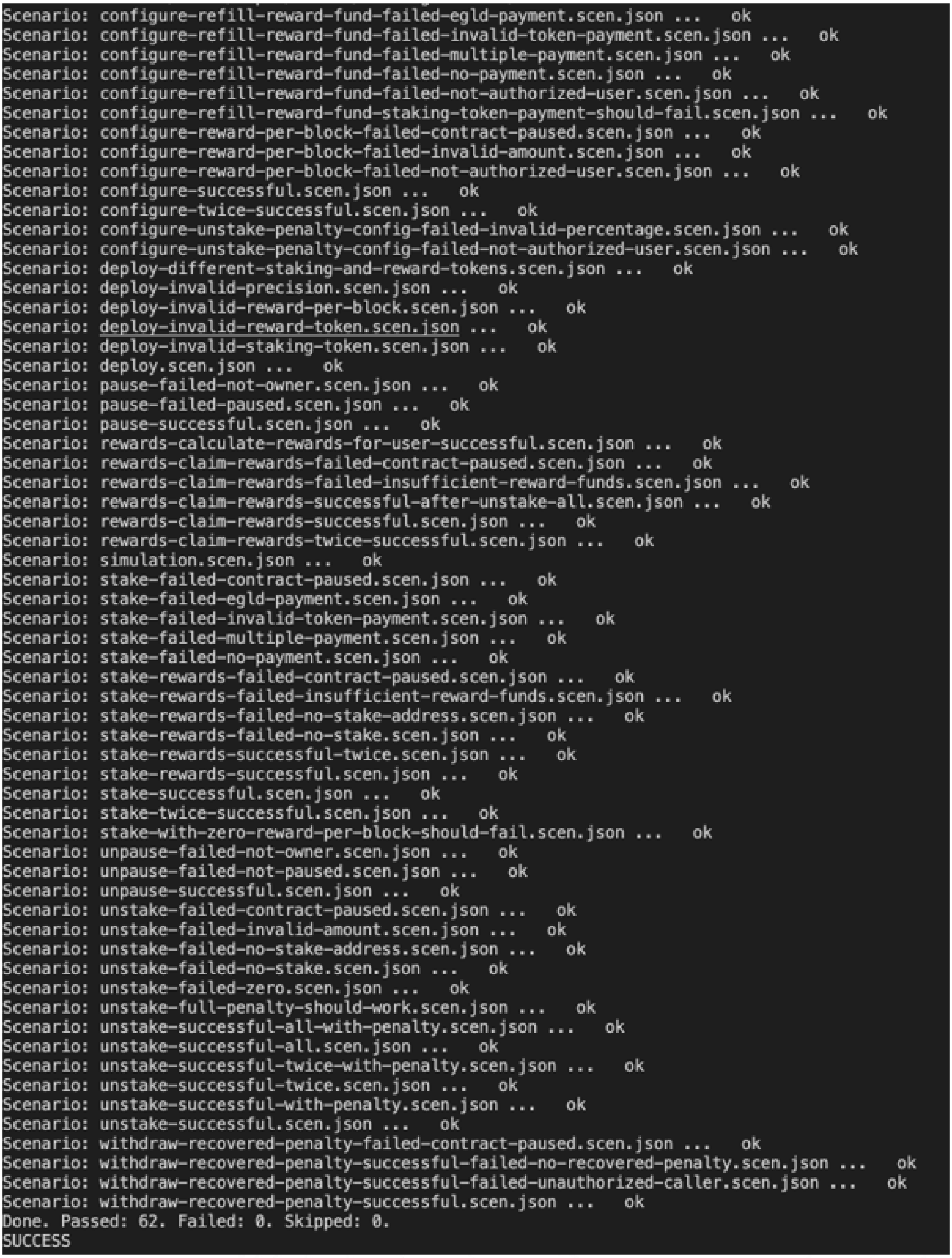
```
Response: Not addressed.

Status: Accepted & Closed (Informational).
```

Test results



```
Scenario: configure-refill-reward-fund-failed-egld-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-invalid-token-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-multiple-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-no-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-not-authorized-user.scen.json ...   ok
Scenario: configure-refill-reward-fund-staking-token-payment-should-fail.scen.json ...   FAIL: result message mismatch. Tx 'step-configure-reward-fund-staking-token-payment-sh
ould-fail'. Want: &{str:Invalid esdt token}. Have:
Scenario: configure-reward-per-block-failed-invalid-amount.scen.json ...   ok
Scenario: configure-reward-per-block-failed-not-authorized-user.scen.json ...   ok
Scenario: configure-successful.scen.json ...   ok
Scenario: configure-twice-successful.scen.json ...   ok
Scenario: configure-unstake-penalty-config-failed-invalid-percentage.scen.json ...   ok
Scenario: configure-unstake-penalty-config-failed-not-authorized-user.scen.json ...   ok
Scenario: deploy-different-staking-and-reward-tokens.scen.json ...   ok
Scenario: deploy-invalid-reward-token.scen.json ...   ok
Scenario: deploy-invalid-staking-token.scen.json ...   ok
Scenario: deploy.scen.json ...   ok
Scenario: rewards-calculate-rewards-for-user-successful.scen.json ...   ok
Scenario: rewards-claim-rewards-failed-insufficient-reward-funds.scen.json ...   ok
Scenario: rewards-claim-rewards-successful-after-unstake-all.scen.json ...   ok
Scenario: rewards-claim-rewards-successful.scen.json ...   ok
Scenario: rewards-claim-rewards-twice-successful.scen.json ...   ok
Scenario: simulation.scen.json ...   ok
Scenario: stake-failed-egld-payment.scen.json ...   ok
Scenario: stake-failed-invalid-token-payment.scen.json ...   ok
Scenario: stake-failed-multiple-payment.scen.json ...   ok
Scenario: stake-failed-no-payment.scen.json ...   ok
Scenario: stake-rewards-failed-insufficient-reward-funds.scen.json ...   ok
Scenario: stake-rewards-failed-no-stake-address.scen.json ...   ok
Scenario: stake-rewards-failed-no-stake.scen.json ...   ok
Scenario: stake-rewards-successful-twice.scen.json ...   ok
Scenario: stake-rewards-successful.scen.json ...   ok
Scenario: stake-successful.scen.json ...   ok
Scenario: stake-twice-successful.scen.json ...   ok
Scenario: stake-with-zero-reward-per-block-should-fail.scen.json ...   FAIL: result code mismatch. Tx 'stake'. Want: 4. Have: 0 (ok). Message:
Scenario: unstake-failed-invalid-amount.scen.json ...   ok
Scenario: unstake-failed-no-stake-address.scen.json ...   ok
Scenario: unstake-failed-no-stake.scen.json ...   ok
Scenario: unstake-failed-zero.scen.json ...   ok
Scenario: unstake-full-penalty-should-work.scen.json ...   FAIL: result code mismatch. Tx 'unstake-successful'. Want: . Have: 10 (execution failed). Message: negative value
Scenario: unstake-successful-all-with-penalty.scen.json ...   ok
Scenario: unstake-successful-all.scen.json ...   ok
Scenario: unstake-successful-twice-with-penalty.scen.json ...   ok
Scenario: unstake-successful-twice.scen.json ...   ok
Scenario: unstake-successful-with-penalty.scen.json ...   ok
Scenario: unstake-successful.scen.json ...   ok
Done. Passed: 42. Failed: 3. Skipped: 0.
ERROR: some tests failed
```

After second review:



```
Scenario: configure-refill-reward-fund-failed-egld-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-invalid-token-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-multiple-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-no-payment.scen.json ...   ok
Scenario: configure-refill-reward-fund-failed-not-authorized-user.scen.json ...   ok
Scenario: configure-refill-reward-fund-staking-token-payment-should-fail.scen.json ...   ok
Scenario: configure-reward-per-block-failed-contract-paused.scen.json ...   ok
Scenario: configure-reward-per-block-failed-invalid-amount.scen.json ...   ok
Scenario: configure-reward-per-block-failed-not-authorized-user.scen.json ...   ok
Scenario: configure-successful.scen.json ...   ok
Scenario: configure-twice-successful.scen.json ...   ok
Scenario: configure-unstake-penalty-config-failed-invalid-percentage.scen.json ...   ok
Scenario: configure-unstake-penalty-config-failed-not-authorized-user.scen.json ...   ok
Scenario: deploy-different-staking-and-reward-tokens.scen.json ...   ok
Scenario: deploy-invalid-precision.scen.json ...   ok
Scenario: deploy-invalid-reward-per-block.scen.json ...   ok
Scenario: deploy-invalid-reward-token.scen.json ...   ok
Scenario: deploy-invalid-staking-token.scen.json ...   ok
Scenario: deploy.scen.json ...   ok
Scenario: pause-failed-not-owner.scen.json ...   ok
Scenario: pause-failed-paused.scen.json ...   ok
Scenario: pause-successful.scen.json ...   ok
Scenario: rewards-calculate-rewards-for-user-successful.scen.json ...   ok
Scenario: rewards-claim-rewards-failed-contract-paused.scen.json ...   ok
Scenario: rewards-claim-rewards-failed-insufficient-reward-funds.scen.json ...   ok
Scenario: rewards-claim-rewards-successful-after-unstake-all.scen.json ...   ok
Scenario: rewards-claim-rewards-successful.scen.json ...   ok
Scenario: rewards-claim-rewards-twice-successful.scen.json ...   ok
Scenario: simulation.scen.json ...   ok
Scenario: stake-failed-contract-paused.scen.json ...   ok
Scenario: stake-failed-egld-payment.scen.json ...   ok
Scenario: stake-failed-invalid-token-payment.scen.json ...   ok
Scenario: stake-failed-multiple-payment.scen.json ...   ok
Scenario: stake-failed-no-payment.scen.json ...   ok
Scenario: stake-rewards-failed-contract-paused.scen.json ...   ok
Scenario: stake-rewards-failed-insufficient-reward-funds.scen.json ...   ok
Scenario: stake-rewards-failed-no-stake-address.scen.json ...   ok
Scenario: stake-rewards-failed-no-stake.scen.json ...   ok
Scenario: stake-rewards-successful-twice.scen.json ...   ok
Scenario: stake-rewards-successful.scen.json ...   ok
Scenario: stake-successful.scen.json ...   ok
Scenario: stake-twice-successful.scen.json ...   ok
Scenario: stake-with-zero-reward-per-block-should-fail.scen.json ...   ok
Scenario: unpause-failed-not-owner.scen.json ...   ok
Scenario: unpause-failed-not-paused.scen.json ...   ok
Scenario: unpause-successful.scen.json ...   ok
Scenario: unstake-failed-contract-paused.scen.json ...   ok
Scenario: unstake-failed-invalid-amount.scen.json ...   ok
Scenario: unstake-failed-no-stake-address.scen.json ...   ok
Scenario: unstake-failed-no-stake.scen.json ...   ok
Scenario: unstake-failed-zero.scen.json ...   ok
Scenario: unstake-full-penalty-should-work.scen.json ...   ok
Scenario: unstake-successful-all-with-penalty.scen.json ...   ok
Scenario: unstake-successful-all.scen.json ...   ok
Scenario: unstake-successful-twice-with-penalty.scen.json ...   ok
Scenario: unstake-successful-twice.scen.json ...   ok
Scenario: unstake-successful-with-penalty.scen.json ...   ok
Scenario: unstake-successful.scen.json ...   ok
Scenario: withdraw-recovered-penalty-failed-contract-paused.scen.json ...   ok
Scenario: withdraw-recovered-penalty-successful-failed-no-recovered-penalty.scen.json ...   ok
Scenario: withdraw-recovered-penalty-successful-failed-unauthorized-caller.scen.json ...   ok
Scenario: withdraw-recovered-penalty-successful.scen.json ...   ok
Done. Passed: 62. Failed: 0. Skipped: 0.
SUCCESS
```

Audited source code version
387649fa04af6bf2be6a53c5d14ed005a3402c64

Second audited source code version
e4b49a864bd4ee3872cb7ea0ace2c2a6ef62222a

*This audited source code version is the latest commit hash of the shared repo.