Debaseonomics - Stabilizer

Security Code Review

https://twitter.com/VidarTheAuditor - 10 December 2020

Overview

Project Summary

Project Name	Debaseonomics
Description	Debaseonomics is a combination of Debase, a flexible supply token, working together with Degov, a governance token working together to solve issues faced by similarly designed tokens. 100% of the tokens are distributed through staking and "stabilizer pools" to promote fairness and decentralization.
Platform	Ethereum, Solidity
Codebase	https://github.com/debaseonomics/stabilizers
Commits	commit 0a2f9cc114d73328c62f4a27f956626b8ca3cccd

Executive Summary

The codebase was found well defined, has proper access restrictions where needed, includes very good comments throughout a code. We have run extensive static analysis of the codebase as well as standard security assessment utilising industry approved tools.

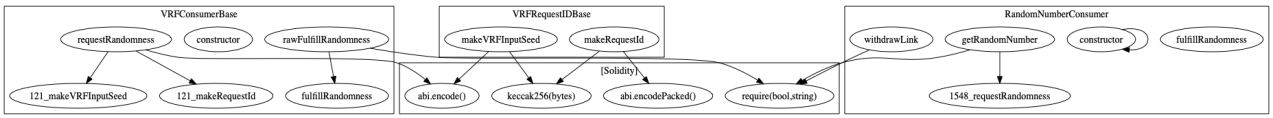
We specially tested a function called from rebase mechanism for any potential issues that could stop rebase.

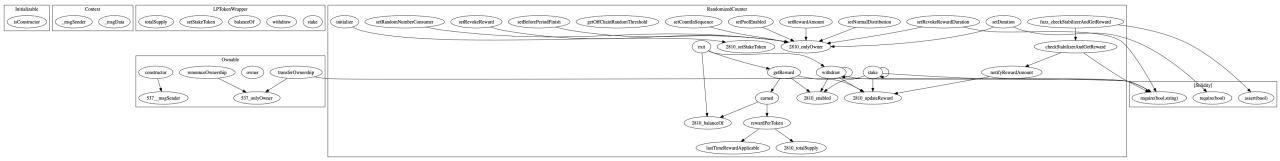
It uses the same pool concept as other Debase pools contracts. It adds the external call to chainlink oracle via interface contract. The oracle is Chainlink VRF providing random number.

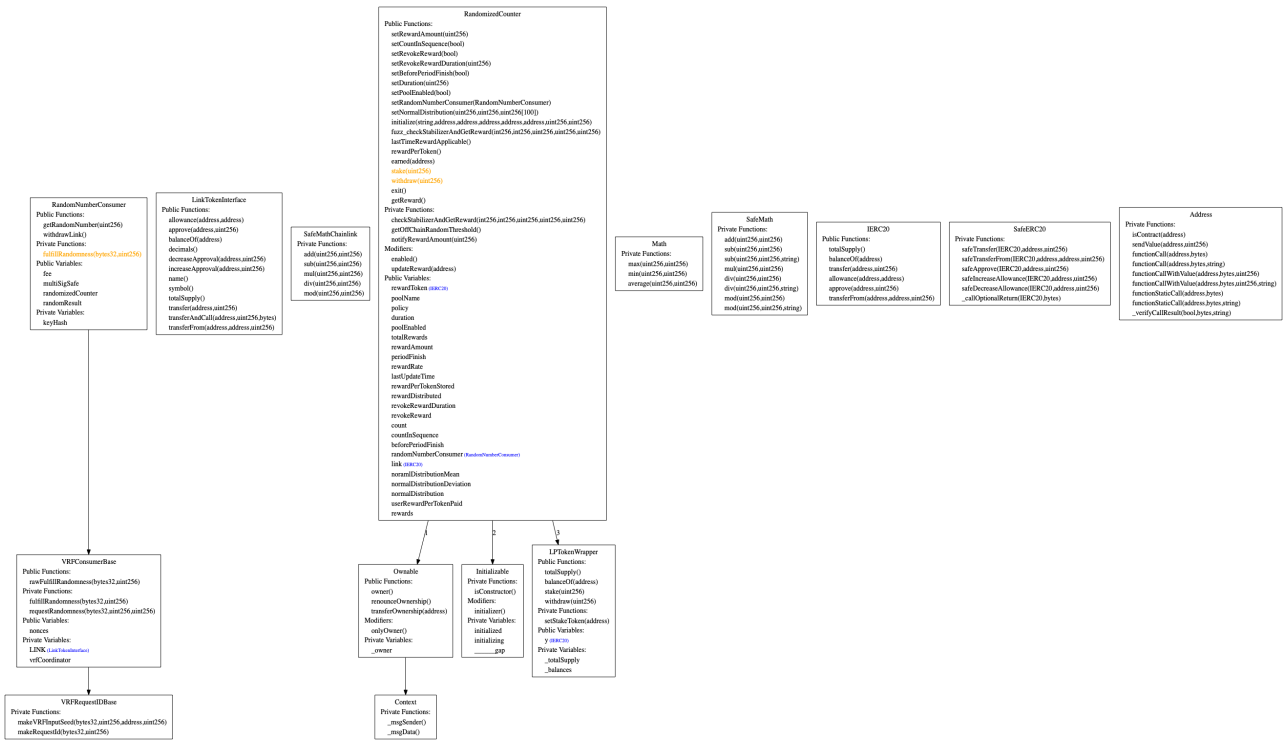
We have found no issues during our review.

Architecture & Standards

Architecture of the stabiliser pool is shown below. It uses the same pool concept as other Debase pools contracts. It adds the external call to chainlink oracle via interface contract. The oracle is Chainlink VRF providing random number.







Findings

Number of contracts: 3 (+ 0 in dependencies, + 0 tests)

Number of assembly lines: 0

Use: Openzeppelin-Ownable, Openzeppelin-SafeMath

Name	# functions	ERCS	ERC20 info	Complex code	Features
RandomNumberConsumer	10			No	Tokens interaction
RandomizedCounter	33			Yes	Send ETH
					Tokens interaction
					Assembly
					Upgradeable

Static Analysis Findings

High issues: None

Medium issues:

Divide before multiply:

```
RandomizedCounter.notifyRewardAmount(uint256) (contracts/Randomized-Threshold-Counter/RandomizedCounter.sol#487-501) performs a multiplication on the result of a division:
        -rewardRate = reward.div(duration) (contracts/Randomized-Threshold-Counter/RandomizedCounter.sol#492)
        -leftover = remaining.mul(rewardRate) (contracts/Randomized-Threshold-Counter/RandomizedCounter.sol#495)
```

In general, it's usually a good idea to re-arrange arithmetic to perform multiplication before division, unless the limit of a smaller type makes this dangerous.

[Manual Check] It does not possesses significant risks for the contract.

Low/Informational issues

Public functions that could be declared external:

```
getRandomNumber(uint256) should be declared external:
        - RandomNumberConsumer.getRandomNumber(uint256) (contracts/Randomized-Threshold-Counter/RandomNumberConsumer.sol#39-49)
initialize(string,address,address,address,address,address,uint256,uint256) should be declared external:
        - RandomizedCounter.initialize(string,address,address,address,address,address,uint256,uint256) (contracts/Randomized-Threshold-Counter/RandomizedCounter.sol#276-303)
fuzz_checkStabilizerAndGetReward(int256,int256,uint256,uint256,uint256) should be declared external:
        - RandomizedCounter.fuzz_checkStabilizerAndGetReward(int256,int256,uint256,uint256,uint256) (contracts/Randomized-Threshold-Counter/RandomizedCounter.sol#378-396)
```

Public functions that are never called by the contract should be declared external to save gas.

Dynamic Tests

We have run fuzzing/property-based testing of Ethereum smarts contracts. It was using sophisticated grammar-based fuzzing campaigns based on a contract ABI to falsify user-defined predicates or Solidity assertions.

We use the fuzzing technique to simulate possible inputs to the checkStabilizerAndGetReward function that is called from debase policy including simulating random numbers from chainlink mock.

```
Tests found: 1
Seed: 3200801788533539362
Unique instructions: 225
Unique codehashes: 1
----------------------------------------------------------------Tests---
assertion in fuzz_checkStabilizerAndGetReward: PASSED!
```

We found no high level issues.

Automatic Tests

We have checked the comprehensive test scripts. They validate the functionality of the contracts. All run successfully.

```
Randomized Threshold Counter
  Deploy and Initialize
    Initial settings
      ✓ Counter reward token should be debase
      ✓ Counter pair token should be dai
      ✓ Counter policy should be policy contract
      ✓ Counter reward amount should be correct
      ✓ Counter duration should be correct
      ✓ Counter pool should be disabled
      ✓ Counter count in sequence should be true
      ✓ Counter initial count should be zero
      ✓ Counter revoke reward should be disabled
      ✓ Counter reward before period finished be disabled
  Check Stabilizer And Get Reward Function
    When supply delta is less than or equal to zero
      ✓ Pool period finish should not change
      ✓ Count should not increase
      ✓ Total rewards should not decrease
    When supply delta is greater than zero
      For single transaction
        ✓ Pool period finish should not change
        ✓ Count should not increase
        ✓ Total rewards should not decrease
      Sequence counter check
        Sequence flag enabled
          ✓ Count in sequence flag should be true
          ✓ Count should be 1 when supply delta is greater than zero
          ✓ Count should reset to 0 when supply delta <= zero (69ms)
        Sequence flag disabled
          ✓ Count in sequence flag should be true
          ✓ Count should be 1 when supply delta is greater than zero
          ✓ Count should be 1 when supply delta <= zero
      For normal mean of 8 and div 0
        For 8 function calls and stabilizer balance less request amount
          ✓ Count before 8th call should be 7
          Call function 8th time
            ✓ Pool period finish eq to zero
            ✓ Total rewards should not decrease
            ✓ Count after 8th call should be 0
            ✓ Total reward amount should not increase
        For 8 function calls and stabilizer balance is more request amount
          ✓ Count before 8th call should be 7
          Call function 8th time
            ✓ Pool period finish not eq to zero
            ✓ Total rewards should not decrease
            ✓ Count after 8th call should be 0
      Revoke reward check
        Revoke reward enabled
          When no rewards available to be revoked
            ✓ Pool period should not change
            ✓ Total rewards should not decrease
          When rewards available to be revoked
            When revoke reward duration is <= period finish
              ✓ Pool period should decrease by revoke reward duration
              ✓ Total rewards should decrease by revoke reward amount
            When revoke reward duration is > period finish
              ✓ Pool period should decrease by revoke reward duration
              ✓ Total rewards should decrease by reward amount


37 passing (19s)
```

Deployment & Contract Ownership

The contracts are not deployed yet.

They should be deployed within current security context including multi-sig wallet and governance structure.

[Recommendation] As debase system is functioning, it is advisable to test the new pool especially checkStabilizerAndGetReward funcionality after deployment before disabling full access by governance system.

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, the author assume no responsibility for errors, omissions, or for damages resulting from the use of the provided information. We do not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One review on its own is not enough for a project to be considered secure; that state can only be earned through extensive peer review and extensive testing over an extended period of time.

The information appearing in this report is for general discussion purposes only and is not intended to provide legal security guarantees to any individual or entity.