StrataLabs

Clarity Contract Breakdown

Prepared For: **Velar**

Latest Review: **Jul. 18th, 23'**

Latest Shared Commit: **Jul. 7th, 23'**

Review II

core.clar & staking-core.clar

Below is a summary & list of all priority (0-3) issues found throughout an independent or paired review session. P0 & P1 issues *must* be fixed prior to launching, p2 issues are highly recommend & p3 issues are optional (usually syntax/preference based)

This section is not periodically updated but another review will be attached below. The top right items detail the date & GitHub commit reviewed.

Issue	Effect	Priority	Category	Status
Overuse of unwrap-panic & try	Possible ambiguous error-handling	**p3**	Inconvenience	Accepted
Single admin	Possibility of lost private keys for admin functions	**p3**	Convention	Accepted
Serial 'And' conditions in asserts	Possible ambiguous error-handling	**p3**	Inconvenience	Accepted

This was our second paired review & light audit for the Velar team for specifically their upcoming **core.clar** & **staking-core.clar** contracts (we ***did*** not audit additional contracts in their protocol). As discussed by the team & seen throughout the contracts, Velar is modeled off of Uniswap v2.

Our contract kicked off on June 22nd with our first preliminary review presented on July 6th; this pass consisted of 104 unit tests written across both contracts. From that initial finding there were no p0 nor p1 as well.

Below are both summaries followed by respective user stories / unit tests.

The .core contract has three (3) main public functions callable by any user: mint which is used to add liquidity to a pool, burn which is used to remove liquidity from a pool & swap, which swaps one token for the other token on the pool making sure that k stays tolerably equal or bigger than a * b to ensure constant product per Uniswap design.

There are additional public functions only callable by the contract owner, which make this contract secure from manipulation. These functions are, set-owner, set-rev-share, set-fee-to & create (used to create a pool pair).

The only other public function in the contract, collect, is only callable by the fee-to address which send all the revenue to the fee-to address.

The staking-core contract has two (2) public functions callable by any user, stake & unstake. It is a relatively straight-forward & standard staking contract that uses maps to track total staked & staked by user. The only token that can be staked is Velar. The user can unstake & stake whenever he/she wants, & everything is tracked in time by epochs which have a duration of 200 blocks.

User Stories - Core.clar

Below is a list of user stories, the corresponding functions & the active state of that story as of the lastest audit.

We can Swap	(swap)	✓	***Can not burn 0 tokens***	(err-burn-preconditions)	✓
Swap must meet minimum token in return	(err-swap-postconditions)	✓	***Burn transfers tokens from protocol to the user***	(burn)	✓
Insufficient balance error for failed trades	(err u1)	✓	***Burns lp-token from the user***	(burn)	✓
Unsupported asset causes trade failure	(err-swap-preconditions)	✓	***Update pool balances after burn***	(update-reserves)	✓
Transfers the fees from the protocol to the rev-share	(swap)	✓	***Transfers the token-in from the user to protocol***	(swap)	✓
Update pool balances post-trade	(update-reserves)	✓	***Transfers the token-out from the protocol to user***	(swap)	✓
Update revenue balances post-trade	(update-revenue)	✓	***Only fee address can call Collect***	(check-fee-to)	✓
We can Mint	(mint)	✓	***Unsupported assets cause collect to fail***	(err-collect-preconditions)	✓
Unsupported asset causes mint to fail	(err-mint-preconditions)	✓	***Transfers amt0 & amt1 from the protocol to fee-to address***	(withdraw-liquidity)	✓
Transfer liquidity to the protocol	(mint)	✓	***Resets revenue to 0***	(reset-revenue)	✓
Transfers lp-token to the liquidity provider	(mint)	✓	***Only owner can change owner***	(check-owner)	✓
Update pool balances post-mint	(update-reserves)	✓	***Only owner can change fee-to address***	(check-owner)	✓
Calculates liquidity from reserves	(calc-mint)	✓	***Only owner can change rev-share address***	(check-owner)	✓
Can not add 0 liquidity	(err-mint-preconditions)	✓	***Fees can only be updated by owner***	(check-owner)	✓
Can not swap 0	(err-swap-preconditions)	✓	***Only owner can create a pool***	(check-owner)	✓
We can Burn	(Burn)	✓	***Can not use a lp-token in 2 pools***	(err-create-preconditions)	✓
Unsupported asset causes burn to fail	(err-burn-preconditions)	✓	***Can not set fee-share lower than 995/1000***	(check-fee)	✓

We can get contract owner	(get-owner)	✓	***Owner can not use the same lp-token in 2 pools***	(err-create-preconditions)	✓
We can get fee-to address	(get-fee-to)	✓	***Owner can not create a pool with wrong fees***	(err-create-preconditions)	✓
We can get rev-share address	(get-rev-share)	✓	***Can not burn if not enough tokens to burn***	(burn)	✓
We can get the number of pools	(get-nr-pools)	✓	***We can get pool information***	(get-pool)	✓

We can get the id of a pool	(get-pool-id)	✓	***Owner can not create a 2nd pool with same tokens***	(err-create-preconditions)	✓
We can get the revenue from a pool	(do-get-revenue)	✓	***Can not change owner if you are not current owner***	(check-owner)	✓
Can not change fee-to if you are not current owner	(check-owner)	✓	***Can not change rev-share if you are not current owner***	(check-owner)	✓
Can not create a pool if you are not owner	(check-owner)	✓	***Swap will fail if fee is bigger than guard***	(check-fee)	✓
Token-0 and token-1 can not be the same	(err-create-preconditions)	✓	***Num can not be bigger than den on fees***	(err-create-preconditions)	✓

User Stories - Staking-Core.clar

Below is a list of user stories, the corresponding functions & the active state of that story as of the lastest audit.

We can Stake	(stake)	✓	***We can get user staked in last interacted epoch***	(get-user-staked)	✓
We can not stake 0	(err-stake-preconditions)	✓	***We can not stake if not enough balance***	(stake)	✓
We can not stake less the 500 the first time	(err-stake-preconditions)	✓	***We can not unstake if not enough staked***	(unstake)	✓

Transfers Velar from user to protocol	(stake)	✓
Updates the current epoch or new epoch stakes total	(if (is-eq epoch...	✓
We can Unstake	(unstake)	✓
Updates the current epoch or new epoch stakes user	(if (is-eq epoch...	✓
We can get share-at specific epoch	(get-share-at)	✓
We can get the current epoch number	(current-epoch)	✓
We can get the block end of an epoch	(calc-epoch-end)	✓

Updates the current epoch or new epoch stakes user	(if (is-eq epoch...	✓
We can not unstake 0	(err-unstake-preconditions)	✓
Can only unstake if u-end1 is 0 or higher than MIN-STAKE	(err-unstake-preconditions)	✓
Updates the current epoch or new epoch stakes total	(if (is-eq epoch...	✓
Transfers Velar from protocol to user	(unstake)	✓
We can get the block start of an epoch	(calc-epoch-start)	✓
We can get total staked in last interacted epoch	(get-total-staked)	✓