Stackswap Audit Report

Tintash

November 10, 2021

Tintash

Table of Contents

1 Report Details

- Prepared For: Sungmin Aum - aum@lucidefi.ai
- Version: git commit - `e0d1638f7f70743cfa9c3970de55ca94ea794eed`
- Reviewers:
  - Faried Nawaz - faried@tintash.com
  - Saad Shafiq - saad.shafiq@tintash.com

2 Summary

During the months of September and October 2021, Stackswap engaged with Tintash to review their smart contracts before their service went live. The contracts include

- DAO and governance contracts
- Several SIP-10 compliant tokens
- Liquidity pool and token swap contracts

Tintash conducted the review over a period of three weeks, with two engineers working on commits `#d82f273` and later `#e0d1638`. The code base contained several older versions of contracts that were identified and not checked for problems.

We made a first pass going over all the contracts to try and gain an understanding of the system, and then deeper passes covering everything from issues with the syntax and use of extraneous/dead code to trying to find weak spots in the interactions between the contracts. Lastly, though not required as part of the review, we looked into superficial ways to cut down transaction execution costs.

During the course of our review, Stackswap implemented suggestions from another reviewer which helped clean up and simplify the code. However, this left some test cases related to the DAO and governance contracts broken. We provided a patch to get things working again.

We did not find any major issues with the contracts. There were a few edge-cases where the code could do a little more sanity checking of the contract call inputs, but nothing critical.

We found areas where improving the test cases to check returned values would lead to better testing of the smart contracts; some tests cases passed properly, though for the wrong reasons.

As Clarity is a relatively new language, there are no tools available to perform detailed semantic analysis of code. In particular, common tasks like identifying unused function parameters and internal variables, and identifying private functions and variables that are never used in a contract, have to be done manually. While a few issues were found of this nature, they were all low priority ones: no cases were found where such code could result in runtime flaws.

At the time of writing this report, all suggested Improvements and fixes are implemented in Stackswap commit `#056057eb`.

Name	Stackswap
Version	e0d1638
Type	Clarity
Platform	Stacks
Method	Whitebox
Reviewers	2

3 Findings

Recommendation	Type	Severity
Governance - block-height should be checked for voting	Bug	Low
Governance - votes and tokens count	Improvement	-
One step mint - use already written functions	Improvement	-
Stackswap v1 - Manual fee re-calculation	Improvement	-
vSTSW - User id for staking	Improvement	-
Poxlsoft v2 - Unused principal argument in poxlv2	Improvement	-
Poxlsoft v2 - divide by zero	Bug	Low
General readability and cleanup - utility function	Improvement	-
Test cases - changes in test cases	Improvement	-

The system was also audited against recent vulnerabilities found in Arkadiko Swap[1], but no issues were found.

4 Recommendations

4.1 Test cases

- All the test cases should check the values returned by contract calls. They should follow an `.expectOk()` or `.expectErr()` with a check for the `value` they should see, using something like `.expectBool(true)` or `.expectUint(4127)`. For example `tests/governance_test.ts` appears to work properly, but this is only because the error condition returned by the function call is not checked; at least one of the calls should fail because of `ERR_BLOCK_PASSED`, but actually fails with `ERR_NOT_ENOUGH_BALANCE`.
- The SIP-10 governance test cases are not properly updated to work properly, but this is only because the error condition returned by the function call is the new `vSTSW` token instead of `STSW`
- Changes in tests are provided as separate file `tests.diff`

4.2 General readability and cleanup

- Several contract functions start off with a lengthy block of code to ensure the callers are valid. Instead of inlining the checks, we recommend that they be separated out into private functions.
- A few functions in several contracts have a `begin` where they don't need one: either the function body contains just one expression, or the `begin` is inside a `let` expression.
- `stake-tokens` in `stackswap-farming.clar` sets a `user` variable but does not use it.
- Usage of fully qualified contract principal is not needed when deployed by same principal.
- Whitespaces, tabs and comments add up in cost calculation. Unused or commented code should be removed and code comments should be minimum.
- Many contracts have duplicates in project, with names suffixed as `-bad` or version numbers.

4.3 Smart Contracts

4.3.1 Governance

Multiple Governance Tokens

`is-token-accepted` in the governance contract only accepts `vSTSW` tokens instead of either `STSW` or `vSTSW`; this makes many test cases fail because the accounts do not have `vSTSW` minted.

[1]https://arkadikofinance.medium.com/arkadiko-swap-post-mortem-f38cef95ff28

Recommendation

Update test cases to use `vSTSW` and make sure that `STSW` is not used in newer versions.

Possible Governance Contract Modification

An unlikely way to break the system was found by voting for a proposal to change the `governance` contract to an invalid principal. After discussion with Stackswap, it appears that this is not possible due to how DAO voting is conducted.

Ending Block Check for Voting

The voting functions do not check `block-height` against the proposal's `end-block-height` value. In the unlikely case that the `(end-proposal)` function is not called (Stackswap's cron jobs do not fire or their servers are unreachable), some votes can be made after voting on the proposal should have ended.

Recommendation

Check for `end-block-height` in voting functions to ensure that voting period is active.

Calling `end-proposal`

`end-proposal` is a public function which can be called by anyone with a valid `proposal-id`.

Recommendation

This should be guarded by `is-council` or `proposer`.

Miscellaneous

- The `votes-by-member` and `tokens-by-member` maps can be merged. The two fields (`vote-count` or `amount`) are redundant and one can be removed.
- `proposer` is not used in `proposals` map. Is this for web only?
- `BASIC_PRINCIPAL` is used as default `proposer`. This can be `optional` and set to `none`. But obviously this will need unwraping when used.
- Error name `ERR_PROPOSAL_IS_NOT_OPEN` assertion against `false` value, is not clearly readable.
- Since `is-open` is a `bool` value so

  ```
      (asserts! (is-eq (get is-open proposal) false) (err
          ERR_PROPOSAL_IS_NOT_OPEN))
  ```

  can be written as

  ```
      (asserts! (get is-open proposal) (err ERR_PROPOSAL_IS_NOT_OPEN))
  ```

4.3.2 One Step Mint

Redfinition, Private and Public functions

There are a few private `remove-token-inner` functions that do the same work as the public `remove-token` functions, except for the `tx-sender` check. The public ones look like they can reuse the private ones.

Recommendation

For example `remove-soft-token` can use private `remove-soft-token-inner` function.

Miscellaneous

The `remove-all` functions can simply do

```
    (var-set soft-token-list (list))
```

instead of using the `empty-token-list` variable (slightly lowering runtime cost this way).

4.3.3 Stackswap V1

Merge to Update Tuples

There are several cases where the `lp-data` tuple can be updated by merging new tuple values on top of an existing `lp-data` instead of creating a new tuple.

Recommendation

An example of this could be

```
   (unwrap! (contract-call? token-liquidity-trait set-lp-data (merge pair {
     fee-to-address: address
   }) token-x token-y) ERR_LP_DATA_SET)
```

Unused `get-fees`

The `swap-x-for-y` and `swap-y-for-x` do not take advantage of the new `get-fees` function in `stackswap-swap-fee-v1.clar`.

Recommendation

Either use `get-fees` in `swap-*-for-*` functions or remove unused `stackswap-swap-fee-v1` contract.

Possible Duplication

In following function

- `add-to-position`
- `reduce-position`
- `swap-x-for-y`
- `swap-y-for-x`
- `collect-fees`

`let` block is used for transaction verification. This block is doing same calculations for almost all variables.

Recommendation

This can be moved in a private function to avoid duplication. While making this change take care of differences like `transfer-*-result`.

4.3.4 vSTSW token

Staking Info

A user can call`stake-tokens` several times, and there's no way for them to later query the contract (using `get-user-info`) for their details of their staking call.

Recommendation

It might be helpful if `stake-tokens` function returns the `user-idx` value at the end of the function.

Unused Parameters

Parameter `idx: uint` in `UserStakingCount` map is not used in contract.

Recommendation

If `idx: uint` is removed then also remove `merge` and use simple `map-set` in `reclaim-token` fucntion for `UserStakingCount` map.

Staking for zero Months

In function `stake-tokens` it is possible to set `month` as `u0`

Recommendation

Add validation for zero value.

4.3.5 Poxl Soft V2

Unused Principal

The `claim-mining-reward-at-block`, `set-mining-reward-claimed`, and `mint-coinbase` functions make use of a principal as a function argument. But it is eventually not used in the called `mint-coinbase` function; that uses `tx-sender` directly.

Recommendation

Use passed down principal argument in `mint-coinbase` funciton.

Divide by Zero

The `reward-cycle-lengh-to-set` value is not checked for to see if it's greater than `u0` when the token is initialized. This can lead to a division-by-zero runtime error in other functions like `get-reward-cycle`.

Recommendation

Check `reward-cycle-length-to-set` value range

Possible Integer Overflow

`get-random-uint-at-block` reads the `vrf-seed` of given block number. The lower 16 bytes are converted to little endian and then to a uint value. Little endian conversion is performed by calling `buff-to-uint-le`, which uses `add-and-shift-uint-le`. For certain values e.g. `acc: u0, data: 0xffffffffffffffffffffffffffffffff` this function will return `acc` larger than size of uint.

Recommendation

Although this is an unlikely value for `VRF`, but further evaluation is required for range verification.

4.3.6 Common Recommendations

Use `contract-caller` to guard crucial funcitons instead of `tx-sender`. This is best explained here[2].

[2]https://app.sigle.io/friedger.id/HuOT9tNQC8fTXOsK28D7e