hTON (Hipo staking protocol) audit report

Invalid unless signed with GPG key ID 40C77B53E493FB74

Performed by ProgramCrafter (manual review) on 2024-02-12 .. 2024-03-06

Results

The code is well commented and tested. Tact-like approach to “routing” (finding function which processes the message) is used, which can increase gas but ensures that each function is called on the corresponding incoming opcode. Methods don't call others (except utility functions) which simplifies validating control flow. The contract system is designed to work under assumption of 32-bit time but has ability to migrate to newer versions.

Most contracts, including money-containing treasury, can be upgraded with `set_code` usage. This functionality is suitably protected.

There were nine findings (1 critical, 1 major, 3 medium, 3 minor, 1 informational), one of which was explained to be an implementation detail, and others fixed promptly. I think Hipo protocol hosted at `https://github.com/HipoFinance/contract/commit/128be3e8be59db9d3c504a17eadcb765ff11a618` is safe!

Findings list

HSP-01, level: critical; fixed in commit 4c13c95.

treasury.fc#L470-476. If there remain no more coins, then tokens are zeroed out but total_unstaking is not. I feel like this can lead to issues.

```
total_coins -= coins;
total_tokens -= tokens;
total_unstaking -= tokens;
if (total_coins == 0) | (total_tokens == 0) {
    total_coins = 0;
    total_tokens = 0;
}
```

HSP-02, level: informational; fixed in commit 882ba1b.

TEP-85 calls SBT destruction by central authority "revocation". I guess bills should use that, including recording the revocation time (for stats how long does it take to wait till [un]staking is complete).

HSP-03, level: medium; fixed in commit fa8fb4b.

Bill doesn't check if it's active when receiving burn request. This means, if it ever gets two burn requests, it's going to respond twice; I'm not sure if that's going to work correctly with other contracts, and this particular check is rather cheap to add.

HSP-04, level: minor; fixed in commit 41c5d88.

Jetton wallet doesn't validate that forward_payload is in a reference if bit in Either is 1 (line 38). It would be better to replace `s~skip_bits(1);` with `throw_unless(err::invalid_parameters, s.slice_refs() >= s~load_uint(1));` or something like that.

HSP-05, level: medium; fixed in commit 4c11341.

Jetton wallet doesn't check if recipient is in basechain; that isn't issue by itself, but then wallet doesn't use recipient workchain when calculating forward fee (line 47).

HSP-06, level: medium; objection withdrawn.

In jetton wallet, checks on lines 48-50 do not exactly make sense. L49-50 check that wallet's TON balance is at least value of incoming message, but since credit phase is before computation one, that's equivalent to "storage fee hasn't sent contract into negative". Also, those checks don't allow owner to pull TON from the jetton wallet if there is some excess. I suggest to replace L48-50 with `int enough_fee? = ton_balance >= (min_ton + wallet_storage_fee());`

HSP-07, level: minor; fixed in commit 8fd6af0.

Jetton wallet doesn't allow to send 0 tokens, throwing an error "err::insufficient_funds". This can be used to initialize jetton wallet for some other contract, actually.

HSP-08, level: major; fixed in commit 90c6a85.

Jetton reception can fail on action phase (meaning no bounce) if destination jetton wallet has a large storage fee due. Forward notification, sent on L94 without "send::ignore_errors", doesn't seem as important as consistency of jettons total amount.

HSP-09, level: minor; fixed in commit ddcf7ed.

I suggest indicating relationships between token variables in TL-B scheme.

```
treasury_storage#_
    total_coins:Coins
    total_tokens:Coins   {  total_coins >= total_tokens  }
    total_staking:Coins
    total_unstaking:Coins
    total_validators_stake:Coins
    parent:MsgAddress
    participations:(HashmapE 32 Participation)
    rounds_imbalance:uint8
    stopped:Bool
    instant_mint:Bool
    loan_codes:(Hashmap 32 ^Cell)
    extension:^Extension    = TreasuryStorage;
```

More precise assertions

1. Loan contract works as intended, only passing money to treasury or Elector.
2. Wallet contract is compliant with jettons standard TEP-74.
3. Bill SBT contract is a functional persistent by-user storage.
4. Bills collection contract is a functional thing allowing one-by-one bills mint and burn. Though, its image for bills is not yet uploaded.
5. “Parent” contract is TEP-89 and TEP-74-compliant jetton master, and also forwards messages to and from treasury as expected.
6. Contracts' code can't be updated by third parties.
7. Likelihood that total_coins < total_tokens in treasury is approximately zero.
8. Unstake operations are recorded and have priority over loan requests, so funds can't get locked in the protocol.
9. total_staking represents TON loaned for a round before hTON mint. It has no provable relationship with total_coins, except that after protocol's first round it will most certainly be lower.
10. total_unstaking represents hTON burned and waiting for payout. As such, total_unstaking <= total_tokens.
11. Operations by one user do not hinder deposits and withdrawals of others.
12. If borrowers-validators don't lose all protocol money (and they likely can't since fines are paid from their money first), users can expect that they can withdraw money without loss in one or two days.