## Osmosis Phase1 Audit

### Scope:

#### Artifacts

* osmosis/cosmos-sdk bank hooks: [link](https://github.com/osmosis-labs/cosmos-sdk/tree/sunny/bank-hooks) - > **MERGED to main**, so main is used in the audit! [6fcd25f](https://github.com/osmosis-labs/cosmos-sdk/commit/6fcd25f8abb82ac4db337d72510429568362d348)
* New features: <https://github.com/osmosis-labs/osmosis/tree/fullpowered-tokenfactory>
* Example of SC implementing hook: <https://github.com/osmosis-labs/cw-usdc>

#### New Token Factory features

* Mint to any account should be safe. Do not analyze this now.
* Larger purpose: in a lot of modules there is an assumption that coins can disappear only if they are transferred. They can not magically disappear.
  + The IBC module is also important! We do not know how IBC behaves!
  + Admin of the token decides to burn all the tokens from IBC.
  + Modules need to be designed to fail gracefully - when tokens balances remove
  + Module accounts are important to focus on!
  + In osmosis we do not have any contracts deployed.

**P1: Before Send hook:** F.E.Distribution module has a lot of internal sending implemented.During this send function if sending of one denom (coins are used to pay tx fees) does not work because the contract (set as before send hook) has freezed the sending - module should continue with distributing funds.

Distribution module executes in end blocker- we do not want it to panic!

We should check if there is any implication that if you have coins in your account you will be able to send them!

Real cases: We have USDC in the distrib module account because USDC was used to pay transaction fees - admin of USDC decides to blacklist the distr module account.

CW20 Standard will be: to implement the Sudo BeforeSendMsg message - it should contain the logic (any logic) when the sending should be enabled and then it should return the error message and stop the execution of the Send Message!

Sudo msg - should probably be using gas - expectations to propose to them how much gas, in order to stop DoS attacks.

**P2: Burn and Force transfer:** Concern that module has the local copy of the amount and does the computation on that information, not on the global state (?) while the global state is changed with burning or forced transfer.

#### Token Factory - code analysis

\*Not discussed at this kick off meeting. Impact is the focus, for now. Update from the sync meeting 02.09.2022: leave for later phase of audit.

### Project Plan

#### Timeline

#### P1 Before Send Hook

* Should be analyzed in **~one month** (end of august) - due to Osmosis Lab agreements to push these changes to main with Circle, for USDC.
* ~~Todo Mirel:~~ give back more details on timing after initial analysis, due to vacation plans. - Info sheridan first sync meeting: planned to wrap up the P1: before send hook analysis until 09/09

#### Tasks:

* List all the modules that could be affected. Analyze Cosmos SDK and osmosis fork’s module for differences. Staking module has a lot of internal sending, but we suppose it should not be affected, since it is dealing with staking OSMO tokens - not under the TokenFactory.
* Wiring and logic within app.go - module’s ordering.
* Analyze osmosis Governance, Distribution, Staking modules and draw diagrams depending on send -ing tokens of certain denominations.
* Focus on different expectations in system design when working with module accounts or with smart contract accounts.
* IBC impact
* Smart Contract implementing Before Send Hook
  + Logic is not important and it is not the subject of auditing
  + Give recommendations when it comes to gas usage - in order to prevent DoS attacks, in case of loops
  + Think of general malicious behavior that could impact the system. What could the smart contract do to halt the chain
* Another idea would be to implement SendBypassHook in the bank module, that would be used for sending from module accounts, not from smart contracts. This would complicate smart contracts development.

#### P2: Burn from and ForceTransfer

* Not that important - the matter of the UX.
* Circle can go without this.

#### Tasks:

* Analyze if there is any module that works with local copy - modules should always use the most recent state (Mirel, comment: not sure if this is an actual issue, but we should check, or ask them)
* Analyze if any of the modules have implicit assumptions that the only way the balance amount could be changed is if the amount is transferred.