Interchain Token Service

Audited by Yaar Hahn
Date: 28/08/2023
Below are a few findings collected in a short audit of the `interchain-token-service` repository.
Commit hash: d743395c171a9a3913c7888996f1eb2b4b1ca438

Findings

- `RemoteAddressValidator.sol` : In case any axelar-enabled chain is compromised it can take over the any ITS, even if ITS wasn't enabled on that chain.
    - Currently `validateSender` doesn't check that the chain is ITS-enabled.
    - If an attacker can compromise one of axelar-enabled chains, and send transactions with a spoofed address, they could send a cross-chain message with the address `interchainTokenServiceAddressHash`. This address is accepted from any chain.
    - I recommend verifying that the chain is ITS-enabled.
- `RemoteAddressValidator.sol` : `getRemoteAddress` doesn't verify that the chain is enabled and fallbacks to the default ITS address.
    - `supportedByGateway` should be queried to make sure the chain is enabled.
    - Might be relevant for other parts of the contract.
- `RemoteAddressValidator.sol`: Missing calls to `addGatewaySupportedChains` in setup.
- `Operatable.sol` - Wrong slot hash for `operator` slot. Should be `0x46a52cf33029de9f84853745a87af28464c80bf0346df1b32e205fc73319f621`.
- `Operatable.sol` - `acceptOperatorship` should zero the `PROPOSED_OPERATOR_SLOT` slot, or else operators can regain operatorship in case of a future `transferOperatorship`.
    - This is a real long shot, but worth the extra slot write just in case.
    - Example: `operator1` proposes `opeartor2` who accepts the operatorship. Later `operator2` transfers operatorship to `operator3`. If `operator2`'s keys are compromised, and calls `acceptOperatorship` again, the operatorship will be stolen.
    - The same issue exists in `Distributable.sol`.
- `InterchainToken.sol` - In the `interchainTransferFrom` function, line 69, the passed sender to `_beforeInterchainTransfer` should be sender and not `msg.sender`.
- `TokenManagerProxy.sol` - The `receive()` function is not needed, as the funds will be stuck forever. Native tokens are only used with `msg.value`, so only the `fallback` function should be accepted, with the `payable` mark (as it is).
- `InterchainTokenService.sol`: In line 643 `getTokenManagerAddress` should be `getValidTokenManagerAddress`.

Informational

- `TokenManager.sol`: `tokenId()` should be `onlyProxy` instead of infinite loop.
- `FlowLimit.sol`: possible gas optimization - no need for a new slot for each epoch. Especially if there's no external view for that (e.g. to query old epochs' volume).
- `TokenManager.sol`: The `setup` function's comment states that "the first 32 bytes are reserved for the address of the operator, stored as bytes (to be compatible with non-EVM chains)". In contrast, the

operator address is casted with `operatorBytes.toAddress()`. `toAddress` verifies that the address is only 20 bytes long.

- It should be noted that in this case, the operator address must be an EVM address so it makes sense that it's limited in size.
- There are no other places in the code that `toAddress` is used on possible non-EVM addresses.