Quicksilver code audit

▼ Table of contents

- ~~Asset/qAsset ratio computation lacks special-case handling: HIGH:: FIXED~~
- Undocumented clever code in ApplyDeltasToIntent: LOW
- x/interchainstaking/types:
  - ~~Missing range check for weight in ConvertMemoToOrdinalIntents: HIGH:: FIXED~~
  - ~~Possibly missing weight validation in IntentsFromString: HIGH: FIXED~~
  - ~~Suspicious TODO in MsgRequestRedemption.ValidateBasic: LOW: FIXED~~
  - ~~Brittle use of NewDecFromStr: LOW: FIXED~~
  - ~~Inconsistent error handling in ConvertMemoToOrdinalIntents: LOW:: FIXED~~
  - ~~indentString could be improved and optimized by using strings.Repeat(indent, n): LOW:: FIXED~~
- x/epochs:
  - x/epochs/types
    - ~~improve the errors for (GenesisState).Validate: LOW:: FIXED~~
- x/epochs/simulation:
  - RandomizedGenesisState is not randomized and is as static as they get: LOW
- x/participationrewards:
  - types:
    - (MsgSubmitClaim) GetSigners() doesn't check that incoming message useraddresses are parseable and catch errors: LOW
    - ~~(MsgSubmitClaim).ValidateBasic is invalid and always returns nil: MEDIUM: FIXED~~
    - ~~(types.AddProtocolDataProposal).ValidateBasic() is return with multiple returns but could simply be 1 line: LOW: FIXED~~
    - ~~(types.Params).Validate() is redundant and can be simplified to just 1 line: LOW:: FIXED~~
    - ~~(types.Params).Validate() and any other method will panic if sent a blank message that'll successfully protobuf unmarshal: CRITICAL: FIXED~~
    - ~~Missing length check in OsmosisPoolUpdateCallback: LOW:: FIXED~~
  - ~~Suspicious code in getRewardsAllocations: LOW:: FIXED~~
  - Token value calculation is incomplete: LOW
  - ~~Incomplete basic validation in SubmitClaim: MEDIUM:: FIXED~~
  - Incomplete protocol data validation in ValidateGenesis: MEDIUM
  - keeper:
    - ~~all codec unmarshallers should firstly check that the data slice is not empty: MEDIUM:: FIXED~~
    - ~~UnmarshalProtocolData can accept blank data {} to parse JSON structs but really for extra checks compare that those values are not equal to the empty value: HIGH:: FIXED~~
    - ~~(keeper.LiquidTokensModule) VerifyClaim doesn't check that the denomination in the claim is the allowed denomination: FALSE-POSITIVE~~
- x/interchainquery:
  - /keeper:
    - Keeper.EndBlocker has a TODO to add the height to the emitted events
  - Keeper.SetAccountBalanceForDenom has more TODOs: MEDIUM
  - /types:
- x/airdrop:
  - ~~keeper: Inconsistent claim record validation: MEDIUM:: FIXED~~
  - keeper: Keeper.decompress is too complex: LOW
- x/mint
- cmd/quicksilverd:
  - ~~AddGenesisAccountCmd has a redundant condition checking if a balance is zero and if the vesting amount is greater but it really should check that the coalesced vesting amounts are not greater than the coalesced balances in case there are duplicate denomination entries in the coins: LOW:: FIXED~~

Preamble:

Customer: Ingenuity dbA Quicksilver

Auditors: Orijtech Inc.

Start date of coverage: Wednesday 1st June 2022

End date of coverage: v1:Saturday 27th August 2022, v1.5:October 31st 2022

Start code SHA commit: 41136a59e0dc0e1e5d07472e805d5944ef326b3b

End code SHA commit: c233ffe23f571a10889270f4124304d49484ebad

Updates made

Version	Start date	Comments
v1	June-1st-2022	Initial version
v1.5	September-10th-2022	Test coverage increases + acknowledgment of the fact that protobuf generated code was heavily skewing coverage results per: https://github.com/ingenuity-build/quicksilver/pull/252 and https://github.com/ingenuity-build/quicksilver/pull/250

Background:

Quicksilver (QCK) a liquid staking protocol whose mission is to efficiently use Cosmos staking liquidity whereby instead of only using 5% staked for security, Quicksilver effectively utilizes 100% by mechanism of issuing an intermediate token. This audit by Orijtech Inc is focused on finding and fixing coding flaws, security issues and vulnerabilities, errors etc. The Quicksilver engineering and management teams were highly responsive and directly addressed each line item identified and reported.

Audited modules:

The critical modules audited were:

- x/participationrewards
- x/interchainstaking
- x/interchainquery
- x/epochs
- x/airdrop
- x/mint
- cmd/quicksilverd

Color coding of found issues:

- CRITICAL
- HIGH
- MEDIUM

- LOW
- NICE TO HAVE
- FALSE POSITIVE
- BLOCKED

Test coverage

Test coverage has been improved lots since v1 of our audit per https://github.com/ingenuity-build/quicksilver/pull/252 and https://github.com/ingenuity-build/quicksilver/pull/250

and notably since v1, we realized that we hadn't accounted for the fact that the repository has heavily generated protobuf code which radically skewed and reported low test coverage.

Despite the big improvements in test coverage, for safety and confidence, our encouragement is for the Quicksilver engineering team to adopt bots like codecov to enforce higher test coverage.

Sparse documentation

Documentation, especially for important state-changing functions and methods, is lacking, but has improved since v1 of our audit.

x/interchainstaking:

Reviewed completely on August 29th 2022.

x/interchainstaking/keeper:

~~(*Keeper).UpdateDelegationRecordsForAddress re-fetching proofs for outdated records happens non-deterministically: HIGH:: FIXED~~

Non-determinism in code can cause issues like much harder consensus due to different validators not agreeing on the same decisions yet running the same code. Please read https://cyber.orijtech.com/cosmos/hardening/

Issue filed https://github.com/ingenuity-build/quicksilver/issues/140

Fix: https://github.com/ingenuity-build/quicksilver/pull/141

~~(msgServer).RequestRedemption redelegation and transaction submission is non-deterministic: HIGH:: FIXED~~

Non-determinism in code can cause issues like much harder consensus due to different validators not agreeing on the same decisions yet running the same code. Please read https://cyber.orijtech.com/cosmos/hardening/

Issue filed https://github.com/ingenuity-build/quicksilver/issues/142

Fix: https://github.com/ingenuity-build/quicksilver/pull/143

~~(ValidatorIntent).Keys performance can be improved by utilizing the size of the source map: LOW & Performance improvement:: FIXED~~

Giving the runtime hints about the map size is important and conserves re-bucketing cycles and memory. Please read https://bencher.orijtech.com/perfclinic/map_copy_capacityhint/

Noticed and reported in https://github.com/ingenuity-build/quicksilver/issues/152

~~AccountBalanceCallback should avoid a runtime panic by invoking sdk.ValidateDenom instead of crashing: LOW:: FIXED~~

Noticed and reported at https://github.com/ingenuity-build/quicksilver/issues/154

This issue is mainly a guard to prevent future misuse and as indicated in the fix and issue discourse, the query was already validated hence the denomination MUST already have been validated. Nonetheless to prevent attacks from higher levels being lux, we should have these guards instead of just crashing.

~~Suspicious code in parseDelegationKey: CRITICAL:: FIXED~~

Filed in https://github.com/ingenuity-build/quicksilver/issues/181 and revealed missing out of bounds of checks which could result in runtime panics due to coding errors

~~Possible missing return in Keeper.BeginBlocker error path: LOW:: FIXED~~

Filed in https://github.com/ingenuity-build/quicksilver/issues/184

~~Possible unbalanced increment/decrement to Zone.WithdrawalWaitgroup: HIGH:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/185

Consider replacing duplicate ibc-go verification code: LOW:: BLOCKED due to dependency issues in ibc-go-v5: ON HOLD

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/186

~~Possible missing out-of-bounds check in AccountBalanceCallback: HIGH:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/187

~~Convoluted code in AccountBalanceCallback: MEDIUM:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/194

~~GetValidatorDelegations should use IterateAllDelegations: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/195

~~Inconsistent error handling in HandleChannelOpenAck: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/196

~~HandleCompleteMultiSend re-implements handleSendToDelegate: MEDIUM:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/197

~~Inconsistent error handling in handleWithdrawForUser: MEDIUM:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/198

~~PrepareRewardsDistributionMsgs ignores rewards denomination: CRITICAL:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/199

~~Needless scaling and truncating of weights in AggregateIntents: HIGH:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/200

~~Error result from GetValidatorByValoper used as found bool: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/201

~~RequestRedemption panics at block height >= 134217728: CRITICAL:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/202

~~Asset/qAsset ratio computation lacks special-case handling: HIGH:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/203

Undocumented clever code in ApplyDeltasToIntent: LOW

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/207

x/interchainstaking/types:

~~Missing range check for weight in ConvertMemoToOrdinalIntents: HIGH:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/183

~~Possibly missing weight validation in IntentsFromString: HIGH: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/216

~~Suspicious TODO in MsgRequestRedemption.ValidateBasic: LOW: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/217

~~Brittle use of NewDecFromStr: LOW: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/218

~~Inconsistent error handling in ConvertMemoToOrdinalIntents: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/182

~~indentString could be improved and optimized by using strings.Repeat(indent, n): LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/153

x/epochs:

x/epochs/types

~~improve the errors for (GenesisState).Validate: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/173

Fixed by PR https://github.com/ingenuity-build/quicksilver/pull/174

x/epochs/simulation:

RandomizedGenesisState is not randomized and is as static as they get: LOW

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/178 but that code is only used in simulation tests,

x/participationrewards:

types:

(MsgSubmitClaim) GetSigners() doesn't check that incoming message useraddresses are parseable and catch errors: LOW

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/208

~~(MsgSubmitClaim).ValidateBasic is invalid and always returns nil: MEDIUM: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/209

~~(types.AddProtocolDataProposal).ValidateBasic() is return with multiple returns but could simply be 1 line: LOW: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/210

~~(types.Params).Validate() is redundant and can be simplified to just 1 line: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/211

~~(types.Params).Validate() and any other method will panic if sent a blank message that'll successfully protobuf unmarshal: CRITICAL: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/212

~~Missing length check in OsmosisPoolUpdateCallback: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/226

~~Suspicious code in getRewardsAllocations: LOW:: FIXED~~

Filed in https://github.com/ingenuity-build/quicksilver/issues/227

Token value calculation is incomplete: LOW

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/228

~~Incomplete basic validation in SubmitClaim: MEDIUM:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/229

Incomplete protocol data validation in ValidateGenesis: MEDIUM

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/232

keeper:

~~all codec unmarshallers should firstly check that the data slice is not empty: MEDIUM:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/213

~~UnmarshalProtocolData can accept blank data `{}` to parse JSON structs but really for extra checks compare that those values are not equal to the empty value: HIGH:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/214

~~(keeper.LiquidTokensModule) VerifyClaim doesn't check that the denomination in the claim is the allowed denomination: FALSE POSITIVE~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/215

x/interchainquery:

/keeper:

Reviewed completely on August 24th 2022 at https://github.com/ingenuity-build/quicksilver/blob/a7f142ab99e978482f66248d0009b90820d0f0ce/x/interchainquery/keeper/

Keeper.EndBlocker has a TODO to add the height to the emitted events

Keeper.SetAccountBalanceForDenom has more TODOs: MEDIUM

https://github.com/ingenuity-build/quicksilver/blob/8c558f01b2b2528bd36c484da2c0b5d64e2b110b/x/interchainstaking/keeper/zones.go#L217-L234

/types:

Completely on August 24th 2022

x/airdrop:

Status: refactored by the Quicksilver team at https://github.com/ingenuity-build/quicksilver/commit/a7f142ab99e978482f66248d0009b90820d0f0ce on August 22nd 2022. Audit completed on August 29th 2022.

~~keeper: Inconsistent claim record validation: MEDIUM:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/219

keeper: Keeper.decompress is too complex: LOW

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/220

x/mint

cmd/quicksilverd:

~~AddGenesisAccountCmd has a redundant condition checking if a balance is zero and if the vesting amount is greater but it really should check that the coalesced vesting amounts are not greater than the coalesced balances in case there are duplicate denomination entries in the coins: LOW:: FIXED~~

Filed in issue https://github.com/ingenuity-build/quicksilver/issues/188

Unfortunately due to a coding error in the cosmos-sdk in which coins cannot be properly coalesced, this problem manifests and is brittle code that assumes only 1 denomination will be added but really a user problem can add more than 1 coin e.g. `{200qck, 200qck}` should be coalesced into `{400qck}` and not naively to be assumed to be `200qck` in 2 different instances. However, that code is only used by the code owners to generate genesis accounts hence a low severity. Reported in the cosmos-sdk per https://github.com/cosmos/cosmos-sdk/issues/13234 and fixed by PR https://github.com/cosmos/cosmos-sdk/pull/13265