HALBORN

Quasar - QBank & QOracle Modules

Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 15th, 2022 - June 26th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/10/2022	Gokberk Gulgun
0.2	Document Updates	06/24/2022	Gokberk Gulgun
0.3	Draft Review	06/25/2022	Gabi Urrutia
1.0	Remediation Plan	07/15/2022	Gokberk Gulgun
1.1	Remediation Plan Review	07/18/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Quasar engaged Halborn to conduct a security assessment on their **Qbank && QOracle** implementation beginning on May 15th and ending on June 26th, 2022.

The security assessment was scoped to the GitHub repository of **Quasar**. An audit of the security risk and implications regarding the changes introduced by the development team at Quasar prior to its production release, shortly following the assessment's deadline.

1.2 AUDIT SUMMARY

The team at Halborn was provided nearly four weeks for the engagement and assigned two full-time security engineers to audit the security of the QBank and QOracle module. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that Quasar QOracle and QBank module functions are intended.
- Identify potential security issues with the Quasar.

In summary, Halborn identified few security risks that were mostly addressed by Quasar Team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the Quasar. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing

techniques help enhance coverage of structures and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (staticcheck, gosec, unconvert, LGTM, ineffassign and semgrep).
- Manual Assessment for discovering security vulnerabilities on code-base.
- Ensuring correctness of the codebase.
- Dynamic Analysis on Quasar QOracle and QBank module functions and data types.
- Property based coverage-guided fuzzing. (gofuzz).

1.4 SCOPE

The assessment was scoped to the repository available in GitHub at commit **aa25077b1ce079d09bf31bc6824f31562acef4cb**.

The audit is only scoped to the following modules :

- QBank.
- QOracle.

FIX Commit Pull Requests :

ICQ Implementation
Pull Request 1
Pull Request 2
Pull Request 3

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	2	4	6	9

EXECUTIVE OVERVIEW

LIKELIHOOD

IMPACT

		(HAL-03)	(HAL-02)	(HAL-01)
	(HAL-04) (HAL-06) (HAL-07)			
(HAL-08) (HAL-10) (HAL-12) (HAL-13)		(HAL-05)		
	(HAL-09) (HAL-11)			
(HAL-14) (HAL-15) (HAL-16) (HAL-17) (HAL-18) (HAL-19) (HAL-20) (HAL-21) (HAL-22)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - PRIVILEGED SINGLE ACCOUNT MANAGES POOL	Critical	SOLVED - 07/18/2022
HAL-02 - THE ORACLE MODULE IS NOT RESISTANT TO DE-PEGGING	High	SOLVED - 07/18/2022
HAL-03 - STABLE DENOM PRICE DIRECTLY CAN BE MANIPULATED ORACLE ACCOUNT	High	SOLVED - 07/18/2022
HAL-04 - POOL INFO MESSAGES ARE NOT VALIDATED THROUGH OSMOSIS	Medium	SOLVED - 07/18/2022
HAL-05 - IMPROPER VALIDATION OF TIMESTAMP	Medium	SOLVED - 07/18/2022
HAL-06 - DENOMS SHOULD NOT BE SAME	Medium	SOLVED - 07/18/2022
HAL-07 - DENOM EXISTING CHECK IS MISSING ON THE PRICE SETTER	Medium	SOLVED - 07/18/2022
HAL-08 - RISK PROFILE IS NOT CONSIDERED ON THE WITHDRAW OPERATIONS	Low	SOLVED - 07/18/2022
HAL-09 - LACK OF SIMULATION AND FUZZING OF QBANK - QORACLE MODULE INVARIANTS	Low	RISK ACCEPTED
HAL-10 - MISSING EVENT PARAMETER ON THE CLAIM REWARDS FUNCTION	Low	SOLVED - 07/18/2022
HAL-11 - INSUFFICIENT VALIDATION OF GENESIS PARAMETERS	Low	RISK ACCEPTED
HAL-12 - LOCK-UP PERIOD IS NOT VALIDATED DURING THE WITHDRAW	Low	SOLVED - 07/18/2022
HAL-13 - ADDTOTALWITHDRAWAMT IS NOT UPDATED DURING THE PARTIAL WITHDRAWAL	Low	SOLVED - 07/18/2022
HAL-14 - MESSAGE VALIDATION CAN BE PLACED INTO VALIDATE BASIC FUNCTION	Informational	SOLVED - 07/18/2022
HAL-15 - LACK OF ERROR HANDLING	Informational	ACKNOWLEDGED

HAL-16 - QBANK DOES NOT TAKE ANY FEE DURING THE DEPOSIT/WITHDRAW	Informational	ACKNOWLEDGED
HAL-17 - MISSING GOLANGCI LINT SUPPORT ON THE REPOSITORY	Informational	SOLVED - 07/18/2022
HAL-18 - ABCI CAN BE REPLACED WITH ABCI++	Informational	ACKNOWLEDGED
HAL-19 - MISSING EMERGENCY PAUSE/UNPAUSE FUNCTIONALITY IN THE QBANK MODULE	Informational	SOLVED - 07/18/2022
HAL-20 - OPEN TODOs	Informational	ACKNOWLEDGED
HAL-21 - REWARDS MAY NOT DISTRIBUTED	Informational	ACKNOWLEDGED
HAL-22 - IOUTIL IS DEPRECATED	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) PRIVILEGED SINGLE ACCOUNT MANAGES POOL - CRITICAL

Description:

QOracle module manages the position of the osmosis pool as present in osmosis DEX. All spot prices are queried from the Osmosis node. However, the oracle module has a single account to manage all pool variables.

Code Location:

OracleAccounts is the function used to authenticate the Oracle Account.

Listing 1

```
    if msg.Creator != k.OracleAccounts(ctx) {
        return nil, types.ErrUnAuthorizedOracleClient
    }
```

This is used as the only authority check in the all pool positions; however, the account is defined as single one on the test cases.

Listing 2: config.yml

```
accounts:
  - name: alice
    mnemonic: edge victory hurry slight dog exit company bike hill erupt shield aspect turkey retreat stairs summer sadness crush absorb draft viable orphan chuckle exhibit
    coins: ["20000token", "200000000stake", "1000000000uqsar"]
  - name: bob
    mnemonic: harvest ill mean warfare gospel slide tragic palace model excess surprise distance voyage change bus grant special artwork win width group dwarf today jar
    coins: ["10000token", "100000000stake", "1000000000uqsar"]
validator:
  name: alice
  staked: "100000000uqsar"
genesis:
```

```
  app_state:
    staking:
      params:
        bond_denom: uqsar
    orion:
      params:
        enabled: false
        lp_epoch_id: minute #override day for testing
        mgmt_fee_per: '0.003000000000000000'
        perf_fee_per: '0.020000000000000000'
    qoracle:
      params:
        oracleAccounts: '
↳ quasar1sqlsc5024sszglyh7pswk5hfpc5xtl77gqjwec'
    qbank:
      params:
        enabled: true
        min_orion_epoch_denom_dollar_deposit: '
↳ 100.000000000000000000'
        orion_epoch_identifier: minute #override day for testing
        white_listed_denoms_in_orion:
          - onehop_osmo: ibc/
↳ BE1BB42D4BE3C30D50B68D7C41DB4DFCE9678E8EF8C539F6E6A9345048894FCC
            onehop_quasar: ibc/
↳ BE1BB42D4BE3C30D50B68D7C41DB4DFCE9678E8EF8C539F6E6A9345048894FCC
            origin_name: uatom
          - onehop_osmo: ibc/
↳ BE1BB42D4BE3C30D50B68D7C41DB4DFCE9678E8EF8C539F6E6A9345048894FCC
            onehop_quasar: uqsar
            origin_name: uqsar
#client:
#  openapi:
#    path: "docs/client/static/openapi/openapi.yml"
#  vuex:
#    path: "vue/src/store"
faucet:
  name: bob
  coins: ["5token", "100000stake", "10000uqsar"]

```

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

Short Term:

- Setup monitoring for activity on the Admin account, any activity should immediately trigger a response.
- Prepare a “break-glass” procedure for the case where this account is compromised. What needs to be done on both chains, who will do it, how to communicate the issue, community interaction. All incident response tasks need to be pre-planned. Assume that it will happen, and design the response to minimize the damage. This procedure will likely involve pausing contracts and halting the oracle nodes to prevent ongoing damage.

Long Term:

- Eliminate the Oracle account.
- Remove any privileged account functions and move these abilities as a function of governance, or delegate the functionality to a super majority vote of either oracles or validators.

Remediation Plan:

SOLVED: The Quasar team states that ICQ implementation will be used. The following implementation was reviewed, and the issue was marked as solved.

3.2 (HAL-02) THE ORACLE MODULE IS NOT RESISTANT TO DE-PEGGING - HIGH

Description:

With the recent problem of UST, the market can be volatile with the stable coins. The current price feeder does not have any protection mechanism for the de-pegging operations. If the providers are feeding with volatile (drop) price, that can cause liquidations. If the price feeder is utilizing a stable coin like UST, the votes should be converted to real USD value before submitting on chain. The many protocols are suffered from the related attack on their system. The example can be seen from the Link.

Even if external oracles like a **Chainlink** are used, during the de-pegging many protocols are affected by the wrong price feed and this caused unintended behavior on the protocols. From that reason, the price feeders should be designed with the real USD value of the assets.

Risk Level:

Likelihood - 4
Impact - 5

Recommendation:

It is recommended to feed prices through real USD value. During the recent market conditions, If the price is feed with stable coin prices, that can directly affect all modules.

Remediation Plan:

SOLVED: To protect protocol from the de-pegging, the Quasar team implemented Band Protocol integration. The issue was solved in PR 89 and PR

104.

3.3 (HAL-03) STABLE DENOM PRICE DIRECTLY CAN BE MANIPULATED ORACLE ACCOUNT - HIGH

Description:

StableDenoms is a list of IBC stable denoms present in the osmosis DEX or any other DEX in the future. This is used to calculate the current market value of any other denoms. On the QOracle module, the stable price is directly set through **MsgStablePrice** message. If the oracle account is compromised, the price can be manipulated and can be feed to other modules.

Code Location:

Listing 3: x/qoracle/keeper/msg_server_stable_price.go

```
1 func (k msgServer) StablePrice(goCtx context.Context, msg *types.
↳ MsgStablePrice) (*types.MsgStablePriceResponse, error) {
2     ctx := sdk.UnwrapSDKContext(goCtx)
3
4     _, err := sdk.AccAddressFromBech32(msg.Creator)
5     if err != nil {
6         return nil, err
7     }
8
9     if msg.Creator != k.OracleAccounts(ctx) {
10        return nil, types.ErrUnAuthorizedOracleClient
11    }
12
13    price := sdk.MustNewDecFromStr(msg.Price)
14    if price.IsNil() || price.IsNegative() {
15        return nil, types.ErrInvalidStablePrice
16    }
17    // AUDIT TODO : oracle account validation to be added.
18
19    k.SetStablePrice(ctx, msg.Denom, price)
20
21    return &types.MsgStablePriceResponse{}, nil
```

```
22 }
23
```

Test:

Listing 4

```
 1 func createStablePrice(k *keeper.Keeper, ctx sdk.Context)
↳ DenomPrice {
 2     price, _ := sdk.NewDecFromStr("10.12")
 3     dp := DenomPrice{Denom: "testd_enom_1", Price: price}
 4
 5     k.SetStablePrice(ctx, dp.Denom, dp.Price)
 6
 7     return dp
 8 }
 9
10 func TestStablePrice(t *testing.T) {
11     setup := testutil.NewTestSetup(t)
12     ctx, k := setup.Ctx, setup.Keepers.QoracleKeeper
13     // Input
14     inputDP1 := createStablePrice(&k, ctx)
15     inputDPS := []DenomPrice{inputDP1}
16
17     // Outputs
18     price1, found := k.GetStablePrice(ctx, inputDP1.Denom)
19     require.True(t, found)
20     var outputDPS []DenomPrice
21     outputDPS = append(outputDPS, DenomPrice{Denom: inputDP1.Denom
↳ , Price: price1})
22     require.ElementsMatch(t,
23         nullify.Fill(inputDPS),
24         nullify.Fill(outputDPS),
25     )
26 }
27
```

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

Do not rely on the spot price to calculate the price of any asset. The price validation should be completed through an external oracle.

Remediation Plan:

SOLVED: Instead of spot prices, the Quasar team implemented Band Protocol integration. The issue was solved in PR 89 and PR 104.

3.4 (HAL-04) POOL INFO MESSAGES ARE NOT VALIDATED THROUGH OSMOSIS - MEDIUM

Description:

Qoracle maintain the state of osmosis pool data as state variables. On the Pool Info messages, the Pool Info is not validated through Osmosis. Even if the messages are privileged through oracle accounts, the invalid data could cause the unexpected behavior on the Qbank.

Code Location:

Listing 5: x/qoracle/types/messages_pool_info.go

```
func (msg *MsgCreatePoolInfo) ValidateBasic() error {
    _, err := sdk.AccAddressFromBech32(msg.Creator)
    if err != nil {
        return sdkerrors.Wrapf(sdkerrors.ErrInvalidAddress, "
↳ invalid creator address (%s)", err)
    }
    if len(msg.PoolId) == 0 {
        return sdkerrors.Wrap(sdkerrors.ErrInvalidRequest, "empty
↳ PoolId")
    }
    if msg.Info == nil {
        return sdkerrors.Wrap(sdkerrors.ErrInvalidRequest, "nil
↳ Info")
    }
    // TODO: Update this with latest osmosis APIs
    // if err := msg.Info.Validate(); err != nil {
    //  return sdkerrors.Wrapf(sdkerrors.ErrInvalidRequest, "
↳ invalid Info (%s)", err)
    // }
    if msg.LastUpdatedTime == 0 {
        return sdkerrors.Wrap(sdkerrors.ErrInvalidRequest, "
↳ LastUpdatedTime is zero")
    }
    return nil
```

```
20 }
```

Test:

Listing 6

```
 1 func sampleBalancerPool() (res gammbalancer.Pool) {
 2     res.Address = "unvalidData"
 3     res.Id = 1
 4     res.PoolParams = gammbalancer.PoolParams{
 5         SwapFee: sdk.NewDecWithPrec(1, 2),
 6         ExitFee: sdk.NewDecWithPrec(1, 2),
 7     }
 8     res.FuturePoolGovernor = "1h"
 9     res.TotalShares = sdk.NewCoin(gammtypes.GetPoolShareDenom(res.
↳ Id), sdk.ZeroInt())
10     res.PoolAssets = []gammtypes.PoolAsset{
11         {
12             Weight: sdk.NewInt(100).MulRaw(gammtypes.
↳ GuaranteedWeightPrecision),
13             Token:  sdk.NewCoin("test", sdk.NewInt(1000)),
14         },
15         {
16             Weight: sdk.NewInt(100).MulRaw(gammtypes.
↳ GuaranteedWeightPrecision),
17             Token:  sdk.NewCoin("test2", sdk.NewInt(100)),
18         },
19     }
20     gammtypes.SortPoolAssetsByDenom(res.PoolAssets)
21     res.TotalWeight = sdk.ZeroInt()
22     for _, asset := range res.PoolAssets {
23         res.TotalWeight = res.TotalWeight.Add(asset.Weight)
24     }
25
26     return
27 }
28
29 func TestMsgCreatePoolInfo_ValidateBasic(t *testing.T) {
30     validPool := sampleBalancerPool()
31
32     tests := []struct {
33         name string
```

```
34          msg  MsgCreatePoolInfo
35          err  error
36      }{
37          {
38              name: "valid",
39              msg: MsgCreatePoolInfo{
40                  Creator:         sample.AccAddressStr(),
41                  PoolId:          "1",
42                  Info:            &validPool,
43                  LastUpdatedTime: 1,
44              },
45          },
46      }
47      for _, tt := range tests {
48          t.Run(tt.name, func(t *testing.T) {
49              err := tt.msg.ValidateBasic()
50              if tt.err != nil {
51                  require.ErrorIs(t, err, tt.err)
52                  return
53              }
54              require.NoError(t, err)
55          })
56      }
57  }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

It is recommended to validate all pool messages through Osmosis API.

Remediation Plan:

SOLVED: The Quasar team states that ICQ implementation will be used. The following implementation was reviewed, and the issue was marked as solved.

3.5 (HAL-05) IMPROPER VALIDATION OF TIMESTAMP - MEDIUM

Description:

During the code review, it has been observed that the timestamp is not validated in all the messages. Qoracle supports the transaction messages for broadcasting the pool info, pool positions, pool spot prices and pool ranking. Price timestamp validation is completed through **LastUpdatedTime** variable. The missing validation can cause stale price. The stale price is unlikely to remain stale as the market absorbs the information and reflects it in the price.

Code Location:

Listing 7: x/qoracle/types

```
func (msg *MsgCreatePoolSpotPrice) ValidateBasic() error {
    _, err := sdk.AccAddressFromBech32(msg.Creator)
    if err != nil {
        return sdkerrors.Wrapf(sdkerrors.ErrInvalidAddress, "
↳ invalid creator address (%s)", err)
    }
    if len(msg.PoolId) == 0 {
        return sdkerrors.Wrap(sdkerrors.ErrInvalidRequest, "empty
↳ PoolId")
    }
    if err := sdk.ValidateDenom(msg.DenomIn); err != nil {
        return sdkerrors.Wrapf(sdkerrors.ErrInvalidRequest, "
↳ invalid DenomIn '%s': %s", msg.DenomIn, err.Error())
    }
    if err := sdk.ValidateDenom(msg.DenomOut); err != nil {
        return sdkerrors.Wrapf(sdkerrors.ErrInvalidRequest, "
↳ invalid DenomOut '%s': %s", msg.DenomOut, err.Error())
    }
    if price, err := sdk.NewDecFromStr(msg.Price); err != nil {
        return sdkerrors.Wrapf(sdkerrors.ErrInvalidRequest, "
↳ invalid Price '%s': %s", msg.Price, err.Error())
    } else if !price.IsPositive() {
```

```
18          return sdkerrors.Wrapf(sdkerrors.ErrInvalidRequest, "Price
↳  '%s' must be positive", msg.Price)
19      }
20      if msg.LastUpdatedTime == 0 {
21          return sdkerrors.Wrap(sdkerrors.ErrInvalidRequest, "
↳ LastUpdatedTime is zero")
22      }
23      return nil
24  }
```

Test:

Listing 8

```
1  func TestMsgUpdatePoolPosition_ValidateBasic(t *testing.T) {
2      samplePoolMetricsMap := createSamplePoolMetricsMap()
3
4      tests := []struct {
5          name string
6          msg  MsgUpdatePoolPosition
7          err  error
8      }{ {
9              name: "zero LastUpdatedTime",
10             msg: MsgUpdatePoolPosition{
11                 Creator: sample.AccAddressStr(),
12                 PoolId:  "1",
13                 Metrics: samplePoolMetricsMap["valid"],
14                 LastUpdatedTime: 1,
15
16             },
17             err: sdkerrors.ErrInvalidRequest,
18         },
19     }
20     for _, tt := range tests {
21         t.Run(tt.name, func(t *testing.T) {
22             err := tt.msg.ValidateBasic()
23             if tt.err != nil {
24                 require.ErrorIs(t, err, tt.err)
25                 return
26             }
27             require.NoError(t, err)
28         })
```

```
29      }
30  }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Ensure that **LastUpdatedTime** is validated according to the latest Unix time in all the messages.

Remediation Plan:

SOLVED: The Quasar team states that ICQ implementation will be used. The following implementation was reviewed, and the issue was marked as solved.

3.6 (HAL-06) DENOMS SHOULD NOT BE SAME - MEDIUM

Description:

QOracle module updates spot prices through CreatePoolSpotPrice function. By using the same denom for both **DenomIn** and **DenomOut**, the spot/stable price can be inflated through QBank module. Therefore, DenomIn and DenomOut should be different.

Code Location:

Listing 9: x/qoracle/keeper/msg_pool_spot_price.go

```
func (k msgServer) CreatePoolSpotPrice(goCtx context.Context, msg
↳ *types.MsgCreatePoolSpotPrice) (*types.
↳ MsgCreatePoolSpotPriceResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    if msg.Creator != k.OracleAccounts(ctx) {
        return nil, types.ErrUnAuthorizedOracleClient
    }

    // Check if the value already exists
    _, isFound := k.GetPoolSpotPrice(
        ctx,
        msg.PoolId,
        msg.DenomIn,
        msg.DenomOut,
    )
    if isFound {
        return nil, sdkerrors.Wrap(sdkerrors.ErrInvalidRequest, "
↳ index already set")
    }

    var poolSpotPrice = types.PoolSpotPrice{
        Creator:          msg.Creator,
        PoolId:           msg.PoolId,
        DenomIn:          msg.DenomIn,
        DenomOut:         msg.DenomOut,
```

```
24          Price:           msg.Price,
25          LastUpdatedTime: msg.LastUpdatedTime,
26      }
27
28      k.SetPoolSpotPrice(
29          ctx,
30          poolSpotPrice,
31      )
32
33      // Storing the stable price of a given input denom
34      // Note - checking only the msg.DenomOut for stable USD denom;
↳  if it is then we are sure
35      // that price of msg.DenomIn is the stable price
36
37      stabledenoms := k.StableDenoms(ctx)
38      for _, stableDenom := range stabledenoms {
39          if msg.DenomOut == stableDenom {
40              decPrice, err := sdk.NewDecFromStr(msg.Price)
41              if err != nil {
42                  panic(err)
43              }
44              k.SetStablePrice(ctx, msg.DenomIn, decPrice)
45          } else if msg.DenomIn == stableDenom {
46              decPrice, err := sdk.NewDecFromStr(msg.Price)
47              if err != nil {
48                  panic(err)
49              }
50              stableDecPrice := sdk.NewDec(1).Quo(decPrice)
51              k.SetStablePrice(ctx, msg.DenomIn, stableDecPrice)
52          }
53      }
54
55      return &types.MsgCreatePoolSpotPriceResponse{}, nil
56  }
```

Test:

Listing 10

```
func TestMsgCreatePoolSpotPrice_ValidateBasic(t *testing.T) {
    tests := []struct {
        name string
        msg  MsgCreatePoolSpotPrice
        err  error
    }{ {
            name: "valid",
            msg: MsgCreatePoolSpotPrice{
                Creator:         sample.AccAddressStr(),
                PoolId:          "1",
                DenomIn:         "abc",
                DenomOut:        "abc",
                Price:           "1.2",
                LastUpdatedTime: 1,
            },
        },
    }
    for _, tt := range tests {
        t.Run(tt.name, func(t *testing.T) {
            err := tt.msg.ValidateBasic()
            if tt.err != nil {
                require.ErrorIs(t, err, tt.err)
                return
            }
            require.NoError(t, err)
        })
    }
}
```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

Ensure that **DenomIn** and **DenomOut** are different and validate it for all the messages.

Remediation Plan:

SOLVED: The code was updated to perform the correct comparison in PR108.

3.7 (HAL-07) DENOM EXISTING CHECK IS MISSING ON THE PRICE SETTER - MEDIUM

Description:

During the code review, It has been noticed that the stable price can be set through the **MsgStablePrice** message. However, the existing check is missing on the StablePrice function. The function should validate If the denom is listed on the StableDenoms.

Code Location:

Listing 11: x/qoracle/keeper/msg_server_stable_price.go

```
func (k msgServer) StablePrice(goCtx context.Context, msg *types.
↳ MsgStablePrice) (*types.MsgStablePriceResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    _, err := sdk.AccAddressFromBech32(msg.Creator)
    if err != nil {
        return nil, err
    }

    if msg.Creator != k.OracleAccounts(ctx) {
        return nil, types.ErrUnAuthorizedOracleClient
    }

    price := sdk.MustNewDecFromStr(msg.Price)
    if price.IsNil() || price.IsNegative() {
        return nil, types.ErrInvalidStablePrice
    }
    // AUDIT TODO : oracle account validation to be added.

    k.SetStablePrice(ctx, msg.Denom, price)

    return &types.MsgStablePriceResponse{}, nil
}

```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

Ensure that the stable token is listed on the module. The validation can be implemented like a **stabledenoms := k.StableDenoms(ctx)**.

Remediation Plan:

SOLVED: The Quasar team states that with the integration with the band protocol; they are going to use the whitelist in the band configuration. This transaction message will now be removed.

3.8 (HAL-08) RISK PROFILE IS NOT CONSIDERED ON THE WITHDRAW OPERATIONS - LOW

Description:

Withdraw transactions allow users to withdraw the currently available withdrawable funds for a specific vault. However, the risk profile is not considered in the current code base.

Code Location:

Listing 12

```
func (k msgServer) RequestWithdraw(goCtx context.Context, msg *types.MsgRequestWithdraw) (*types.MsgRequestWithdrawResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    depositor := msg.GetCreator()
    coin := msg.GetCoin()
    vaultId := msg.GetVaultID()
    riskProfile := msg.GetRiskProfile()

    depositorAddr, err := sdk.AccAddressFromBech32(depositor)
    if err != nil {
        return nil, err
    }

    switch vaultId {
    case oriontypes.ModuleName:
        wcoin := k.GetActualWithdrawableAmt(ctx, depositor, coin.Denom)
        if wcoin.Amount.LT(coin.Amount) {
            return nil, types.ErrWithdrawInsufficientFunds
        }

        // Transfer amount to depositor from vault module acc.
        err := k.bankKeeper.SendCoinsFromModuleToAccount(
            ctx,
```

```
24              oriontypes.ModuleName,
25              depositorAddr,
26              sdk.NewCoins(coin),
27          )
28          if err != nil {
29              return nil, err
30          }
31
32      default:
33          return nil, types.ErrInvalidVaultId
34      }
35
36      k.Keeper.SubActualWithdrawableAmt(ctx, depositor, coin)
37
38      ctx.EventManager().EmitEvent(
39          types.CreateWithdrawEvent(ctx, depositorAddr, coin,
↳ vaultId, riskProfile),
40      )
41
42      k.Logger(ctx).Info(
43          "RequestWithdraw",
44          "Depositor", depositor,
45          "Coin", coin.String(),
46          "VaultId", vaultId,
47          "RiskProfile", riskProfile,
48      )
49
50      return &types.MsgRequestWithdrawResponse{}, nil
51  }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider review and delete or update functionality on the all unused components.

Remediation Plan:

SOLVED: The risk profile is removed from the module. Instead of this component, the reserved field array has been added for future use in PR108

3.9 (HAL-09) LACK OF SIMULATION AND FUZZING OF QBANK - QORACLE MODULE INVARIANTS - LOW

Description:

The Quasar system lacks comprehensive Cosmos SDK simulations and invariants for its x/qbank and x/qoracle modules. More thorough use of the simulation feature would facilitate fuzz testing of the entire blockchain and help ensure that the invariants hold.

Code Location:

Listing 13: x/qoracle/simulation/stable_price.go

```
func SimulateMsgStablePrice(
    ak types.AccountKeeper,
    bk types.BankKeeper,
    k keeper.Keeper,
) simtypes.Operation {
    return func(r *rand.Rand, app *baseapp.BaseApp, ctx sdk.
↳ Context, accs []simtypes.Account, chainID string,
    ) (simtypes.OperationMsg, []simtypes.FutureOperation, error) {
        simAccount, _ := simtypes.RandomAcc(r, accs)
        msg := &types.MsgStablePrice{
            Creator: simAccount.Address.String(),
        }

        // TODO: Handling the StablePrice simulation

        return simtypes.NoOpMsg(types.ModuleName, msg.Type(), "
↳ StablePrice simulation not implemented"), nil, nil
    }
}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Long term, extend the simulation module to cover all operations that may occur in a real Quasar deployment, along with all potential error states, and run it many times before each release. Ensure the following:

- All modules and operations are included in the simulation module.
- The simulation uses a few accounts (e.g., between 5 and 20) to increase
  the likelihood of an interesting state change.
- The simulation uses the currencies/tokens that will be used in the production network.
- Oracle price changes are properly simulated. (In addition to a mode in which prices are changed randomly, implement a mode in which prices are changed only slightly, a mode in which prices are highly volatile, and a mode in which prices decrease or increase continuously for a long time period.)
- The simulation continues running when a transaction triggers an error.
- All transaction code paths are executed. (Enable code coverage to see how often individual lines are executed.)

Remediation Plan:

RISK ACCEPTED: The Quasar Team accepted the risk of this finding.

3.10 (HAL-10) MISSING EVENT PARAMETER ON THE CLAIM REWARDS FUNCTION - LOW

Description:

Events are objects that contain information about the execution of the application. They are mainly used by service providers like block explorers and wallet to track the execution of various messages and index transactions. In the ClaimRewards function, the claimed amount is not emitted through EventManager.

Code Location:

Listing 14

```
func (k msgServer) ClaimRewards(goCtx context.Context, msg *types.
↳ MsgClaimRewards) (*types.MsgClaimRewardsResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    depositor := msg.GetCreator()
    vaultId := msg.GetVaultID()

    depositorAddr, err := sdk.AccAddressFromBech32(depositor)
    if err != nil {
        return nil, err
    }

    switch vaultId {
    case oriontypes.ModuleName:
        qcoins, found := k.GetUserClaimAmt(ctx, depositor, vaultId
↳ )
        if found {
            rewardAccName := oriontypes.
↳ CreateOrionRewardGloablMaccName()
            err := k.bankKeeper.SendCoinsFromModuleToAccount(
                ctx,
                rewardAccName,
                depositorAddr,
```

```
21                  qcoins.Coins,
22              )
23              if err != nil {
24                  return nil, err
25              }
26
27              k.ClaimAll(ctx, depositor, vaultId)
28              k.AddUserClaimedRewards(ctx, depositor, vaultId,
↳ qcoins.Coins)
29          }
30
31      default:
32          return nil, types.ErrInvalidVaultId
33      }
34
35      ctx.EventManager().EmitEvent(
36          types.CreateClaimRewardsEvent(ctx, depositorAddr, vaultId)
↳ ,
37      )
38
39      k.Logger(ctx).Info(
40          "ClaimRewards",
41          "Depositor", depositor,
42          "VaultId", vaultId,
43      )
44
45      // TODO - Define and Emit Events
46      return &types.MsgClaimRewardsResponse{}, nil
47  }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to omit all related parameters on the events.

Remediation Plan:

SOLVED: The code was updated so that the event was added in PR108.

3.11 (HAL-11) INSUFFICIENT VALIDATION OF GENESIS PARAMETERS - LOW

Description:

A few system parameters must be set correctly for the system to function properly. The system checks the parameter input against minimum and maximum values (not always correctly) but does not check the correctness of the parameters' dependencies. When preparing a protocol upgrade, the Quasar team accidentally introduces an invalid value into the configuration file. As a result, the upgrade is deployed with an invalid or unexpected parameter.

Code Location:

Listing 15: x/qoracle/genesis.go

```
func InitGenesis(ctx sdk.Context, k keeper.Keeper, genState types.
↳ GenesisState) {
    // Set all the poolPosition
    for _, elem := range genState.PoolPositionList {
        k.SetPoolPosition(ctx, elem)
    }
    // Set if defined
    if genState.PoolRanking != nil {
        k.SetPoolRanking(ctx, *genState.PoolRanking)
    }
    // Set all the poolSpotPrice
    for _, elem := range genState.PoolSpotPriceList {
        k.SetPoolSpotPrice(ctx, elem)
    }
    // Set all the poolInfo
    for _, elem := range genState.PoolInfoList {
        k.SetPoolInfo(ctx, elem)
    }
    // this line is used by starport scaffolding # genesis/module/
↳ init
    k.SetParams(ctx, genState.Params)
```

```
20 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to implement proper validation of configurable values to ensure that the following expected invariants.

Remediation Plan:

RISK ACCEPTED: The Quasar team accepted the risk of this finding.

3.12 (HAL-12) LOCK-UP PERIOD IS NOT VALIDATED DURING THE WITHDRAW - LOW

Description:

During the code review, It has been observed that the user can deposit through white-listed lock-up periods. Without checking risk-profile or lock-up period, the user can complete withdraw workflow.

Code Location:

Listing 16: x/qoracle/keeper/msg_server_request_withdraw.go

```
func (k msgServer) RequestWithdraw(goCtx context.Context, msg *
↳ types.MsgRequestWithdraw) (*types.MsgRequestWithdrawResponse,
↳ error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    depositor := msg.GetCreator()
    coin := msg.GetCoin()
    vaultId := msg.GetVaultID()
    riskProfile := msg.GetRiskProfile()

    depositorAddr, err := sdk.AccAddressFromBech32(depositor)
    if err != nil {
        return nil, err
    }

    switch vaultId {
    case oriontypes.ModuleName:
        wcoin := k.GetActualWithdrawableAmt(ctx, depositor, coin.
↳ Denom)
        if wcoin.Amount.LT(coin.Amount) {
            return nil, types.ErrWithdrawInsufficientFunds
        }

        // Transfer amount to depositor from vault module acc.
        err := k.bankKeeper.SendCoinsFromModuleToAccount(
            ctx,
```

```
24              oriontypes.ModuleName,
25              depositorAddr,
26              sdk.NewCoins(coin),
27          )
28          if err != nil {
29              return nil, err
30          }
31
32      default:
33          return nil, types.ErrInvalidVaultId
34      }
35
36      k.Keeper.SubActualWithdrawableAmt(ctx, depositor, coin)
37
38      ctx.EventManager().EmitEvent(
39          types.CreateWithdrawEvent(ctx, depositorAddr, coin,
↳ vaultId, riskProfile),
40      )
41
42      k.Logger(ctx).Info(
43          "RequestWithdraw",
44          "Depositor", depositor,
45          "Coin", coin.String(),
46          "VaultId", vaultId,
47          "RiskProfile", riskProfile,
48      )
49
50      return &types.MsgRequestWithdrawResponse{}, nil
51  }
```

Test:

Listing 17

```
1  func TestRequestWithdraw(t *testing.T) {
2      setup := testutil.NewTestSetup(t)
3      k := setup.Keepers.QbankKeeper
4      userAddr := sample.AccAddress()
5      mintAmount := sdk.NewInt(int64(1000000000))
6      targetAmount := sdk.NewInt(int64(42))
7      server, srvCtx := setupMsgServer(setup.Ctx, k)
8      var err error
```

```
 9
10      // Mint some coins for orion account
11      setup.Keepers.AccountKeeper.NewAccountWithAddress(setup.Ctx,
↳ userAddr)
12      err = setup.Keepers.BankKeeper.MintCoins(
13          setup.Ctx,
14          oriontypes.ModuleName,
15          sdk.NewCoins(sdk.NewCoin("QSR", mintAmount)),
16      )
17      require.NoError(t, err)
18
19      // Give a claim of targetAmount of these coins for a user
20      k.AddActualWithdrawableAmt(setup.Ctx, userAddr.String(), sdk.
↳ NewCoin("QSR", targetAmount))
21
22      // Then withdraw a target amount
23      w := types.NewMsgRequestWithdraw(
24          userAddr.String(),
25          "HIGH",
26          "orion",
27          sdk.NewCoin("QSR", targetAmount),
28      )
29      res, err := server.RequestWithdraw(srvCtx, w)
30      require.NoError(t, err)
31      require.NotNil(t, res)
32
33      ctx := sdk.UnwrapSDKContext(srvCtx)
34      eventtest.AssertEventEmitted(t, ctx, types.TypeEvtWithdraw)
35
36      balance := setup.Keepers.BankKeeper.GetBalance(setup.Ctx,
↳ userAddr, "QSR")
37      require.Equal(t, targetAmount, balance.Amount)
38      require.Equal(t, "QSR", balance.Denom)
39  }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is recommended to implement proper validation of lock-up, however If It does not affect user portfolio in the other modules the workflow should be documented.

Remediation Plan:

RISK ACCEPTED: The Quasar team states that the behavior is as expected according to the current design. Consider the case of the orion module, which first aggregates user deposits in one place and executes the liquidity mining strategy on the osmosis pool. Due to the nature of osmosis pools, they are subject to the risk of impermanent loss. Which may cause the actual amount to be withdrawn to be less than the actual amount deposited. After the lockup period ends and orion exits the osmosis pool. Orion will calculate the amount to be withdrawn and add the withdrawable amount to the qbank withdrwable kv store based on the user's account. This design helps create good segregation and decoupling of the three deposit, withdraw and claim operations, which could be used in multiple types of vaults.

3.13 (HAL-13) ADDTOTALWITHDRAWAMT IS NOT UPDATED DURING THE PARTIAL WITHDRAWAL - LOW

Description:

During the code review, It has been observed that total withdrawn amount is updated through keeper. However, on the partial payments, total withdrawn amount is not updated.

Code Location:

Listing 18: x/qbank/keeper/msg_server_request_withdraw.go

```
func (k msgServer) RequestWithdraw(goCtx context.Context, msg *
↳ types.MsgRequestWithdraw) (*types.MsgRequestWithdrawResponse,
↳ error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    depositor := msg.GetCreator()
    coin := msg.GetCoin()
    vaultId := msg.GetVaultID()
    riskProfile := msg.GetRiskProfile()

    depositorAddr, err := sdk.AccAddressFromBech32(depositor)
    if err != nil {
        return nil, err
    }

    switch vaultId {
    case oriontypes.ModuleName:
        wcoin := k.GetActualWithdrawableAmt(ctx, depositor, coin.
↳ Denom)
        if wcoin.Amount.LT(coin.Amount) {
            return nil, types.ErrWithdrawInsufficientFunds
        }

        // Transfer amount to depositor from vault module acc.
        err := k.bankKeeper.SendCoinsFromModuleToAccount(
            ctx,
```

```
24              oriontypes.ModuleName,
25              depositorAddr,
26              sdk.NewCoins(coin),
27          )
28          if err != nil {
29              return nil, err
30          }
31
32      default:
33          return nil, types.ErrInvalidVaultId
34      }
35
36      k.Keeper.SubActualWithdrawableAmt(ctx, depositor, coin)
37
38      ctx.EventManager().EmitEvent(
39          types.CreateWithdrawEvent(ctx, depositorAddr, coin,
↳ vaultId, riskProfile),
40      )
41
42      k.Logger(ctx).Info(
43          "RequestWithdraw",
44          "Depositor", depositor,
45          "Coin", coin.String(),
46          "VaultId", vaultId,
47          "RiskProfile", riskProfile,
48      )
49
50      return &types.MsgRequestWithdrawResponse{}, nil
51  }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Ensure that all keys are updated correctly.

Remediation Plan:

SOLVED: The Quasar team fixed the issue by adding **AddTotalWithdrawAmt** amount in PR108

3.14 (HAL-14) MESSAGE VALIDATION CAN BE PLACED INTO VALIDATE BASIC FUNCTION - INFORMATIONAL

Description:

ValidateBasic is happening during the CheckTx phase, and it doesn't have access to the state. So, it can verify only the current object. In the current implementation, of the modules, the message parameter validation should be moved into the **ValidateBasic** function to improve readability and consistency.

Code Location:

Listing 19: /x/qoracle/types/message_stable_price.go

```
func (msg *MsgStablePrice) ValidateBasic() error {
    _, err := sdk.AccAddressFromBech32(msg.Creator)
    if err != nil {
        return sdkerrors.Wrapf(sdkerrors.ErrInvalidAddress, "
↳ invalid creator address (%s)", err)
    }
    return nil
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Ensure all validations are moved into ValidateBasic function.

Remediation Plan:

SOLVED: The Quasar team states that they are using stateless validations in basic validation. However, the stable price tx message will be removed after successful completion of the band integration.

3.15 (HAL-15) LACK OF ERROR HANDLING - INFORMATIONAL

Description:

Some sections of the codebase contain calls to functions which may throw errors. Failure to handle error conditions may result in unexpected behavior, information disclosure (such as stack traces), and denial-of-service in the case where the lack of error handling causes a node to crash. Instead of using a panic function, the error should be handled through Cosmos SDK. When the unexpected operation is constructed by a user, the transaction should be reverted.

Code Location:

Listing 20

```
./keeper/withdraw.go:62:        panic(fmt.Errorf("empty
↳ withdrawable amount for key=%v", string(key)))
./keeper/withdraw.go:116:       panic(fmt.Errorf("empty
↳ withdrawable amount for key=%v", string(key)))
./keeper/withdraw.go:171:       panic(fmt.Errorf("empty
↳ withdrawable amount for key=%v", string(key)))
./keeper/claim.go:80:       panic(fmt.Sprintf("claim amount is
↳ empty for the key key=%v", string(key)))
./keeper/stable_price.go:16:        panic(err)
./keeper/msg_server_pool_spot_price.go:52:              panic(err)
./keeper/msg_server_pool_spot_price.go:58:              panic(err)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Ensure that errors are handled properly to avoid any potential security impacts. When writing unit tests, consider adding test cases that include unexpected and invalid input to ensure that a greater ranger of errors is caught.

Remediation Plan:

ACKNOWLEDGED: The Quasar Team acknowledged this issue.

3.16 (HAL-16) QBANK DOES NOT TAKE ANY FEE DURING THE DEPOSIT/WITHDRAW - INFORMATIONAL

Description:

In the Quasar modules, There is no protocol fee taken during the withdraw/deposit functions. Whenever a user withdraws/deposits from a vault, a withdrawal fee is not taken. The protocol should ensure that the flow is on-purpose.

Code Location:

Listing 21

```
func (k msgServer) ClaimRewards(goCtx context.Context, msg *types.
↳ MsgClaimRewards) (*types.MsgClaimRewardsResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    depositor := msg.GetCreator()
    vaultId := msg.GetVaultID()

    depositorAddr, err := sdk.AccAddressFromBech32(depositor)
    if err != nil {
        return nil, err
    }

    switch vaultId {
    case oriontypes.ModuleName:
        qcoins, found := k.GetUserClaimAmt(ctx, depositor, vaultId
↳ )
        if found {
            rewardAccName := oriontypes.
↳ CreateOrionRewardGloablMaccName()
            err := k.bankKeeper.SendCoinsFromModuleToAccount(
                ctx,
                rewardAccName,
                depositorAddr,
                qcoins.Coins,
```

```
22              )
23              if err != nil {
24                  return nil, err
25              }
26
27              k.ClaimAll(ctx, depositor, vaultId)
28              k.AddUserClaimedRewards(ctx, depositor, vaultId,
↳ qcoins.Coins)
29          }
30
31      default:
32          return nil, types.ErrInvalidVaultId
33      }
34
35      ctx.EventManager().EmitEvent(
36          types.CreateClaimRewardsEvent(ctx, depositorAddr, vaultId)
↳ ,
37      )
38
39      k.Logger(ctx).Info(
40          "ClaimRewards",
41          "Depositor", depositor,
42          "VaultId", vaultId,
43      )
44
45      // TODO - Define and Emit Events
46      return &types.MsgClaimRewardsResponse{}, nil
47  }
48
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Ensure that zero fee implementation is intended.

Remediation Plan:

ACKNOWLEDGED: The Quasar team states that the behavior is as expected based on the current design. The fee deduction is delegated to the actual vault in which users deposit through qbank. Qbank facilitate simple deposit, withdrawal and claim operations for the vault. Vault will update the value of the claimable and withdrawable amount only after performance and management fees are deducted.

3.17 (HAL-17) MISSING GOLANGCI LINT SUPPORT ON THE REPOSITORY - INFORMATIONAL

Description:

During the code review, It has been observed there is no GitHub action has been defined for the scanning code base with the code linters.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use https://golangci-lint.run after the every PR/merge. The relevant findings should be fixed before the deployment.

Remediation Plan:

SOLVED: The Quasar team solved the issue by adding GoLint in the pipeline.

3.18 (HAL-18) ABCI CAN BE REPLACED WITH ABCI++ - INFORMATIONAL

Description:

ABCI is the interface between the consensus engine and the application. It defines when the application can talk to consensus during the execution of a blockchain. Currently, the application can only act at one phase in consensus, immediately after a block has been finalized. ABCI can only act in one phase of consensus, immediately after a block has been finalized. This restriction on the application prevents the application from implementing numerous features, including many scalability improvements, that are now better understood than when ABCI was first written. Many of the scalability proposals, for example, boil down to “Make the miners / block proposers / validators do the work, so the network doesnt have to.” Optimizations such as tx-level signature aggregation, state transition proofs, and so on are included. Furthermore, many new security properties are impossible to achieve in the current paradigm because the application cannot compel validators to do more than just finalize txs. Threshold cryptography and guaranteed IBC connection attempts are examples of such features.

ABCI++ overcomes these constraints by allowing the application to intervene at three critical points during block execution. The new interface enables block proposers to perform application-dependent work in a proposed block via the **PrepareProposal** method; validators to perform application-dependent work in a proposed block via the **ProcessProposal** method; and applications to require their validators to do more than just validate blocks via the **ExtendVote** and VerifyVoteExtension methods. Furthermore, **ABCI++** renames **BeginBlock**, **[DeliverTx]**, and **EndBlock** to **FinalizeBlock** to make it easier to deliver a decided block to the Application.

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Halborn recommends that the review the workflow on the ABCI and the take the advantage of **ABCI++**.

Remediation Plan:

ACKNOWLEDGED: The Quasar Team acknowledged this issue.

3.19 (HAL-19) MISSING EMERGENCY PAUSE/UNPAUSE FUNCTIONALITY IN THE QBANK MODULE - INFORMATIONAL

Description:

In case a hack is occuring or an exploit is discovered, the team (or validators in this case) should be able to pause functionality until the necessary changes are made to the system. Because an attack would probably span a number of blocks, a method for pausing the module would be able to interrupt any such attack if discovered. To use a thorchain example again, the team behind thorchain noticed an attack was going to occur well before the system transferred funds to the hacker. However, they were not able to shut the system down fast enough. Incident Report

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Pause functionality on the modules would have helped secure the funds quickly.

Remediation Plan:

SOLVED: The Quasar team states that the enable flag in qbank has already been added, which could be used in the emergency scenario to disable qbank deposits. However, the withdrawal and claim methods are open currently, so the user can still withdraw from their claimable and withdrawable transactions. They could also add enable check in the message server for these two transactions.

3.20 (HAL-20) OPEN TODOs - INFORMATIONAL

Description:

Open TODOs can point to architecture or programming issues that still need to be resolved. For instance, the following code section includes denom white-listing through Orion, however, the hash values will be different at the production.

Code Location:

Listing 22: x/qbank/types/params.go

```
var (
    KeyEnabled                          = []byte("Enabled")
    KeyMinOrionEpochDenomDollarDeposit = []byte("
↳ MinOrionEpochDenomDollarDeposit")
    KeyOrionEpochIdentifier             = []byte("
↳ OrionEpochIdentifier")
    KeyWhiteListedDenomsInOrion         = []byte("
↳ WhiteListedDenomsInOrion")

    DefaultEnabled                                  = false
    DefaultMinOrionEpochDenomDollarDeposit sdk.Dec = sdk.
↳ NewDecWithPrec(100, 0) // 100.0 Dollar
    DefaultOrionEpochIdentifier                     = "day"

    // AUDIT NOTE - Below commented value are used for local
↳ testing -with different values of ibc hexh hash.
    // And should be uncommented in the final production code.
    denom1 WhiteListedDenomInOrion = WhiteListedDenomInOrion{
        OriginName:   "uatom",
        OnehopQuasar: "ibc/
↳ BE1BB42D4BE3C30D50B68D7C41DB4DFCE9678E8EF8C539F6E6A9345048894FCC",
        OnehopOsmo:   "ibc/
↳ BE1BB42D4BE3C30D50B68D7C41DB4DFCE9678E8EF8C539F6E6A9345048894FCC",
    }
    //denom2 WhiteListedDenomInOrion = WhiteListedDenomInOrion{
↳ OriginName: "uosmo", OnehopQuasar: "IBC/TESTQSRATOM", OnehopOsmo:
```

```
↳ "IBC/TESTOSMOOSMO"}
19     //DefaultWhiteListedDenomsInOrion = []WhiteListedDenomInOrion{
↳ denom1, denom2}
20     DefaultWhiteListedDenomsInOrion = []WhiteListedDenomInOrion{
↳ denom1}
21 )
```

TO-DO:

Listing 23

```
1 ./types/genesis.go:6:// TODO | AUDIT | qbank genesis state to be
↳ redefined about the kind of state object/objects it should
2 ./module_simulation.go:30:  // TODO: Determine the simulation
↳ weight value
3 ./module_simulation.go:34:  // TODO: Determine the simulation
↳ weight value
4 ./module_simulation.go:38:  // TODO: Determine the simulation
↳ weight value
5 ./module_simulation.go:42:  // TODO: Determine the simulation
↳ weight value
6 ./spec/06_queries.md:27:TODO - should change the sequence of
↳ arguments.
7 ./simulation/request_withdraw.go:25:        // TODO: Handling the
↳ RequestWithdraw simulation
8 ./simulation/claim_rewards.go:25:       // TODO: Handling the
↳ ClaimRewards simulation
9 ./simulation/request_withdraw_all.go:25:        // TODO: Handling
↳ the RequestWithdrawAll simulation
10 ./keeper/msg_server_request_deposit.go:54:  // TODO AG merge these
↳  3 calls into a single public function in the keeper
11 ./keeper/msg_server_claim_rewards.go:58:    // TODO - Define and
↳ Emit Events
12 ./keeper/epoch.go:8:    // TODO get epoch identifier from params
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Consider resolving the TODOs before the production.

Remediation Plan:

ACKNOWLEDGED: The Quasar Team acknowledged this issue.

3.21 (HAL-21) REWARDS MAY NOT DISTRIBUTED - INFORMATIONAL

Description:

If the qbank module's **rewardAccName** lacks the coins to cover a reward payout, the rewards will not be distributed or registered for payment in the future.

Code Location:

Listing 24: x/qbank/keeper/msg_server_claim_rewards.go (Line 29)

```
25      switch vaultId {
26      case oriontypes.ModuleName:
27          qcoins, found := k.GetUserClaimAmt(ctx, depositor, vaultId
↳ )
28          if found {
29              rewardAccName := oriontypes.
↳ CreateOrionRewardGloablMaccName()
30              err := k.bankKeeper.SendCoinsFromModuleToAccount(
31                  ctx,
32                  rewardAccName,
33                  depositorAddr,
34                  qcoins.Coins,
35              )
36              if err != nil {
37                  return nil, err
38              }
39
40              k.ClaimAll(ctx, depositor, vaultId)
41              k.AddUserClaimedRewards(ctx, depositor, vaultId,
↳ qcoins.Coins)
42          }
43
44      default:
45          return nil, types.ErrInvalidVaultId
46      }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Short term, document the fact that oracle rewards will not be distributed when the **rewardAccName** does not have enough coins to cover the rewards.

Remediation Plan:

ACKNOWLEDGED: The Quasar Team acknowledged this issue.

3.22 (HAL-22) IOUTIL IS DEPRECATED - INFORMATIONAL

Description:

The package **ioutil** is deprecated after Go **1.16**. New code is encouraged to use the respective implementations in the packages **io** and **os**.

Code Location:

Listing 25: x/qoracle/client/cli/parse-pool.go

```
1 func parseBalancerPoolFile(poolFile string) (*gammbalancer.Pool,
↳ error) {
2     contents, err := ioutil.ReadFile(poolFile)
3     if err != nil {
4         return nil, err
5     }
6
7     pool := &gammbalancer.Pool{}
8     err = json.Unmarshal(contents, pool)
9     if err != nil {
10        return nil, err
11    }
12
13    return pool, nil
14 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider changing **ioutil** with **io** library.

Remediation Plan:

ACKNOWLEDGED: The Quasar Team acknowledged this issue.

AUTOMATED TESTING

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec ineffassign, unconvert and LGTM. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.

Semgrep - Security Analysis Output Sample:

Listing 26: Rule Set

```
semgrep --config "p/dgryski.semgrep-go" x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o dgryski.semgrep
semgrep --config "p/owasp-top-ten"      x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o owasp-top-ten.semgrep
semgrep --config "p/r2c-security-audit" x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o r2c-security-audit.semgrep
semgrep --config "p/r2c-ci"             x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o r2c-ci.semgrep
semgrep --config "p/ci"                 x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o ci.semgrep
semgrep --config "p/golang"             x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o golang.semgrep
semgrep --config "p/trailofbits"        x --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o trailofbits.semgrep
```

Semgrep Results:

```
Fetching rules from https://semgrep.dev/registry.
Scanning 63 files with 19 go rules.
  89%|████████████████████████████████████████████          |56/63 tasks
 100%|██████████████████████████████████████████████████████|63/63 tasks

Findings:

  types/query.pb.gw.go
     trailofbits.go.questionable-assignment.questionable-assignment
        Should `protoReq` be modified when an error could be returned?
        Details: https://sg.run/qq6y

         74┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        108┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        142┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        169┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        196┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        223┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        250┆ protoReq.UserAccount, err = runtime.String(val)
          ⋮┆----------------------------------------
        261┆ protoReq.Denom, err = runtime.String(val)
          ⋮┆----------------------------------------
        288┆ protoReq.UserAccount, err = runtime.String(val)
          ⋮┆----------------------------------------
        299┆ protoReq.Denom, err = runtime.String(val)
          ⋮┆----------------------------------------
        326┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        337┆ protoReq.VaultID, err = runtime.String(val)
          ⋮┆----------------------------------------
        364┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        375┆ protoReq.VaultID, err = runtime.String(val)
          ⋮┆----------------------------------------
        402┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        413┆ protoReq.VaultID, err = runtime.String(val)
          ⋮┆----------------------------------------
        440┆ protoReq.UserAcc, err = runtime.String(val)
          ⋮┆----------------------------------------
        451┆ protoReq.VaultID, err = runtime.String(val)
```

Gosec - Security Analysis Output Sample:

```
[gosec] 2022/06/26 16:43:33 Including rules: default
[gosec] 2022/06/26 16:43:33 Excluding rules: default
[gosec] 2022/06/26 16:43:33 Import directory: [redacted]/quasar-217159fb6d739ae7a2b2851c07c834f1205d5836/x/qbank
[gosec] 2022/06/26 16:43:48 Checking package: qbank
[gosec] 2022/06/26 16:43:48 Checking file: [redacted]/quasar-217159fb6d739ae7a2b2851c07c834f1205d5836/x/qbank/genesis.go
[gosec] 2022/06/26 16:43:48 Checking file: [redacted]/quasar-217159fb6d739ae7a2b2851c07c834f1205d5836/x/qbank/handler.go
[gosec] 2022/06/26 16:43:48 Checking file: [redacted]/quasar-217159fb6d739ae7a2b2851c07c834f1205d5836/x/qbank/module.go
[gosec] 2022/06/26 16:43:48 Checking file: [redacted]/quasar-217159fb6d739ae7a2b2851c07c834f1205d5836/x/qbank/module_simulation.go
Results:


[[redacted]/x/qbank/module_simulation.go:41] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   40:
  > 41:         opWeightMsgRequestWithdrawAll = "op_weight_msg_request_withdraw_all"
   42:         // TODO: Determine the simulation weight value


[[redacted]/qbank/module_simulation.go:37] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   36:
  > 37:         opWeightMsgClaimRewards = "op_weight_msg_claim_rewards"
   38:         // TODO: Determine the simulation weight value


[[redacted]/x/qbank/module_simulation.go:33] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   32:
  > 33:         opWeightMsgRequestWithdraw = "op_weight_msg_request_withdraw"
   34:         // TODO: Determine the simulation weight value


[[redacted]/x/qbank/module_simulation.go:29] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   28: const (
  > 29:         opWeightMsgRequestDeposit = "op_weight_msg_request_deposit"
   30:         // TODO: Determine the simulation weight value


Summary:
  Gosec  : 2.11.0
  Files  : 4
  Lines  : 510
  Nosec  : 0
  Issues : 4
[gosec] 2022/06/26 16:47:36 Including rules: default
[gosec] 2022/06/26 16:47:36 Excluding rules: default
[gosec] 2022/06/26 16:47:36 Import directory: [redacted]/x/qoracle
[gosec] 2022/06/26 16:47:37 Checking package: qoracle
[gosec] 2022/06/26 16:47:37 Checking file: [redacted]/x/qoracle/genesis.go
[gosec] 2022/06/26 16:47:37 Checking file: [redacted]/x/qoracle/handler.go
[gosec] 2022/06/26 16:47:37 Checking file: [redacted]/x/qoracle/module.go
[gosec] 2022/06/26 16:47:37 Checking file: [redacted]/x/qoracle/module_simulation.go
Results:


[[redacted]/x/qoracle/module_simulation.go:75] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   74:
  > 75:         opWeightMsgStablePrice = "op_weight_msg_stable_price"
   76:         // TODO: Determine the simulation weight value


[[redacted]/x/qoracle/module_simulation.go:71] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   70:
  > 71:         opWeightMsgDeletePoolInfo = "op_weight_msg_create_chain"
   72:         // TODO: Determine the simulation weight value


[[redacted]6/x/qoracle/module_simulation.go:67] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   66:
  > 67:         opWeightMsgUpdatePoolInfo = "op_weight_msg_create_chain"
   68:         // TODO: Determine the simulation weight value


[[redacted]6/x/qoracle/module_simulation.go:63] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   62:
  > 63:         opWeightMsgCreatePoolInfo = "op_weight_msg_create_chain"
   64:         // TODO: Determine the simulation weight value


[[redacted]/x/qoracle/module_simulation.go:59] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
   58:
  > 59:         opWeightMsgDeletePoolSpotPrice = "op_weight_msg_create_chain"
   60:         // TODO: Determine the simulation weight value
```

THANK YOU FOR CHOOSING

// HALBORN