HALBORN

Persistence - StkATOM

Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 4th, 2022 - October 20th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Edits	10/05/2022	Gokberk Gulgun
0.2	Document Edits	10/08/2022	Gokberk Gulgun
0.3	Draft Review	10/21/2022	Gabi Urrutia
1.0	Remediation Plan	10/24/2022	Gokberk Gulgun
1.1	Remediation Plan Review	10/24/2022	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
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Persistence engaged Halborn to conduct a security audit on their **stkATOM** Implementation, beginning on October 4th, 2022 and ending on October 20th, 2022 . The security assessment was scoped to the code base provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided nearly five weeks for the engagement and assigned two full-time security engineers to audit the security of the **stkATOM** Implementation. 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 stkATOM Implementation functions as intended.
- Identify potential security issues with the Persistence Team.

In summary, Halborn identified few security risks that were mostly addressed by the Persistence 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 **stkATOM** Implementation. 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 codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis on stkATOM Implementation functions and data types.

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating

a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to persistenceOne/pStake-native repository.

Commit ID

IN-SCOPE MODULES :

- x/lscosmos.

FIX COMMIT ID:

FIX Commit ID

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	5	12	6

EXECUTIVE OVERVIEW

LIKELIHOOD

IMPACT

(HAL-06) (HAL-07) (HAL-10) (HAL-11) (HAL-12) (HAL-13) (HAL-14) (HAL-15) (HAL-16) (HAL-17)		(HAL-01) (HAL-02) (HAL-03) (HAL-04) (HAL-05)		
	(HAL-08) (HAL-09)			
(HAL-18) (HAL-19) (HAL-20) (HAL-21) (HAL-22) (HAL-23)				

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - MISSING C-RATE THRESHOLD RATIO	Medium	SOLVED - 10/24/2022
HAL-02 - PROTOCOL FEE DOES NOT HAVE UPPER BOUND	Medium	SOLVED - 10/24/2022
HAL-03 - MISSING FUNCTIONALITY FOR MOVING REMAINING TOKENS TO PSTAKEADDRESS	Medium	SOLVED - 10/05/2022
HAL-04 - NEW HOST CHAIN PROPOSAL PARAMETERS ARE NOT VALIDATED	Medium	SOLVED - 10/24/2022
HAL-05 - NON-DETERMINISTIC ITERATIONS CAN CAUSE CONSENSUS FAILURES	Medium	SOLVED - 10/24/2022
HAL-06 - ONTIMEOUTPACKET METHOD NOT IMPLEMENTED	Low	SOLVED - 10/24/2022
HAL-07 - MUSTACCADDRESSFROMBECH32 CAN LEADS TO PANIC IN THE CHAIN	Low	SOLVED - 10/24/2022
HAL-08 - FEE ADDRESS IS DEFINED AS AN ACCOUNT	Low	RISK ACCEPTED
HAL-09 - LACK OF SIMULATION AND FUZZING OF THE MODULE INVARIANT	Low	RISK ACCEPTED
HAL-10 - LACK OF SPEC ON THE MODULE	Low	RISK ACCEPTED
HAL-11 - COMPUTATIONALLY HEAVY OPERATIONS IN BEGINBLOCKER MAY SLOW DOWN OR STOP BLOCK PRODUCTION	Low	SOLVED - 10/24/2022
HAL-12 - TEST DOCKER IMAGE RUNNING AS ROOT	Low	RISK ACCEPTED
HAL-13 - DELEGATE MESSAGES WITH ZERO AMOUNT IS NOT CHECKED	Low	SOLVED - 10/24/2022
HAL-14 - GENESIS STATE DOES NOT HAVE ENOUGH VALIDATION	Low	SOLVED - 10/24/2022
HAL-15 - CODECS DO NOT HAVE INIT FUNCTION	Low	SOLVED - 10/24/2022

HAL-16 - USE REGISTERLEGACYAMINOCODEC INSTEAD OF REGISTERCODE	Low	SOLVED - 10/24/2022
HAL-17 - CROSS SITE SCRIPTING THROUGH GOVERNANCE PROPOSAL	Low	RISK ACCEPTED
HAL-18 - TYPO ON THE VARIABLES	Informational	SOLVED - 10/24/2022
HAL-19 - OPEN TODOS	Informational	SOLVED - 10/24/2022
HAL-20 - PANIC IS USED FOR ERROR HANDLING	Informational	ACKNOWLEDGED
HAL-21 - RENAME TO THE MODULE	Informational	SOLVED - 10/24/2022
HAL-22 - LSCOSMOS DOES NOT USE REST CLI HANDLER	Informational	ACKNOWLEDGED
HAL-23 - REGISTERGRPCGATEWAYROUTES IS NOT FUNCTIONAL	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) MISSING C-RATE THRESHOLD RATIO - MEDIUM

Description:

GetCValue calculates the C value after every time when the staking/minting is completed through the module. However, the calculation is completed through the function, If the minted amount is much higher than staked amount, the calculation can lead to inconsistency on the module. From that reason, It is recommended to define proper bound on the c value.

Code Location:

/x/lscosmos/keeper/c_value.go, Lines 62

Listing 1

```
func (k Keeper) GetCValue(ctx sdk.Context) sdk.Dec {
    stakedAmount := k.GetDepositAccountAmount(ctx).
        Add(k.GetDelegationAccountAmount(ctx)).
        Add(k.GetIBCTransferTransientAmount(ctx)).
        Add(k.GetDelegationTransientAmount(ctx)).
        Add(k.GetStakedAmount(ctx)).
        Add(k.GetHostDelegationAccountAmount(ctx))

    mintedAmount := k.GetMintedAmount(ctx)
    if stakedAmount.IsZero() || mintedAmount.IsZero() {
        return sdk.OneDec()
    }

    return sdk.NewDecFromInt(mintedAmount).Quo(sdk.NewDecFromInt(
↳ stakedAmount))
}
```

Proof Of Concept:

Listing 2

```
func (suite *IntegrationTestSuite) TestCValue() {
    app, ctx := suite.app, suite.ctx

    lscosmosKeeper := app.LSCosmosKeeper

    amounts := sdk.NewCoins(sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).MintDenom, 1000000))
    err := app.BankKeeper.MintCoins(ctx, types.ModuleName, amounts)
    suite.NoError(err)

    cValue := lscosmosKeeper.GetCValue(ctx)
    suite.Equal(sdk.OneDec(), cValue)

    cValue = lscosmosKeeper.GetCValue(ctx)
    tokenValue, residue := lscosmosKeeper.ConvertStkToToken(ctx, sdk.NewDecCoin(lscosmosKeeper.GetHostChainParams(ctx).MintDenom, sdk.NewInt(1000000)), cValue)
    suite.True(sdk.NewInt64Coin(lscosmosKeeper.GetIBCDenom(ctx), 1000000).IsEqual(tokenValue))
    suite.True(sdk.NewDecCoinFromDec(lscosmosKeeper.GetIBCDenom(ctx), sdk.ZeroDec()).IsEqual(residue))

    cValue = lscosmosKeeper.GetCValue(ctx)
    stkValue, residue := lscosmosKeeper.ConvertTokenToStk(ctx, sdk.NewDecCoin(lscosmosKeeper.GetIBCDenom(ctx), sdk.NewInt(1000000)), cValue)
    suite.True(sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).MintDenom, 1000000).IsEqual(stkValue))
    suite.True(sdk.NewDecCoinFromDec(lscosmosKeeper.GetHostChainParams(ctx).MintDenom, sdk.ZeroDec()).IsEqual(residue))

    supply := lscosmosKeeper.GetMintedAmount(ctx)
    suite.True(amounts.AmountOf(lscosmosKeeper.GetHostChainParams(ctx).MintDenom).Equal(supply))

    delegationState := types.DelegationState{
        HostAccountDelegations: []types.HostAccountDelegation{
            {
                ValidatorAddress: "",
```

```
30                 Amount:           sdk.NewInt64Coin(lscosmosKeeper.
↳ GetHostChainParams(ctx).BaseDenom, 600000),
31             },
32             {
33                 ValidatorAddress: "",
34                 Amount:           sdk.NewInt64Coin(lscosmosKeeper.
↳ GetHostChainParams(ctx).BaseDenom, 200000),
35             },
36             {
37                 ValidatorAddress: "",
38                 Amount:           sdk.NewInt64Coin(lscosmosKeeper.
↳ GetHostChainParams(ctx).BaseDenom, 100000),
39             },
40             {
41                 ValidatorAddress: "",
42                 Amount:           sdk.NewInt64Coin(lscosmosKeeper.
↳ GetHostChainParams(ctx).BaseDenom, 100000),
43             },
44         },
45         HostDelegationAccountBalance: sdk.NewCoins(sdk.
↳ NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).BaseDenom,
↳ 1000)),
46     }
47
48     lscosmosKeeper.SetDelegationState(ctx, delegationState)
49
50     stakedAmount := lscosmosKeeper.GetStakedAmount(ctx)
51     suite.True(sdk.NewInt(1000000).Equal(stakedAmount))
52
53     cValue = lscosmosKeeper.GetCValue(ctx)
54     suite.Equal(sdk.NewDecWithPrec(999000999000999001, 18), cValue
↳ )
55
56     cValue = lscosmosKeeper.GetCValue(ctx)
57     tokenValue, residue = lscosmosKeeper.ConvertStkToToken(ctx,
↳ sdk.NewDecCoin(lscosmosKeeper.GetHostChainParams(ctx).MintDenom,
↳ sdk.NewInt(1000000)), cValue)
58     suite.True(sdk.NewInt64Coin(lscosmosKeeper.GetIBCDenom(ctx),
↳ 1001000).IsEqual(tokenValue))
59     suite.True(sdk.NewDecCoinFromDec(lscosmosKeeper.GetIBCDenom(
↳ ctx), sdk.ZeroDec()).IsEqual(residue))
60
61     cValue = lscosmosKeeper.GetCValue(ctx)
```

```
62      stkValue, residue = lscosmosKeeper.ConvertTokenToStk(ctx, sdk.
↳ NewDecCoin(lscosmosKeeper.GetIBCDenom(ctx), sdk.NewInt(1000000)),
↳ cValue)
63      suite.True(sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(
↳ ctx).MintDenom, 999000).IsEqual(stkValue))
64      suite.True(sdk.NewDecCoinFromDec(lscosmosKeeper.
↳ GetHostChainParams(ctx).MintDenom, sdk.NewDecWithPrec
↳ (999000999001000000, 18)).IsEqual(residue))
65
66      hostChainParams := lscosmosKeeper.GetHostChainParams(ctx)
67      ibcDenom := ibctransfertypes.ParseDenomTrace(
68          ibctransfertypes.GetPrefixedDenom(
69              hostChainParams.TransferPort, hostChainParams.
↳ TransferChannel, hostChainParams.BaseDenom,
70          ),
71      ).IBCDenom()
72
73      err = app.BankKeeper.MintCoins(ctx, types.ModuleName, sdk.
↳ NewCoins(sdk.NewCoin(ibcDenom, sdk.NewInt(20000))))
74      suite.NoError(err)
75
76      err = app.BankKeeper.SendCoinsFromModuleToModule(ctx,
77          types.ModuleName,
78          types.DepositModuleAccount,
79          sdk.NewCoins(sdk.NewCoin(ibcDenom, sdk.NewInt(10000))),
80      )
81      suite.NoError(err)
82
83      err = app.BankKeeper.SendCoinsFromModuleToModule(ctx,
84          types.ModuleName,
85          types.DelegationModuleAccount,
86          sdk.NewCoins(sdk.NewCoin(ibcDenom, sdk.NewInt(10000))),
87      )
88      suite.NoError(err)
89
90      cValue = lscosmosKeeper.GetCValue(ctx)
91      suite.Equal(sdk.NewDecWithPrec(979431929480901077, 18), cValue
↳ )
92
93      cValue = lscosmosKeeper.GetCValue(ctx)
94      tokenValue, residue = lscosmosKeeper.ConvertStkToToken(ctx,
↳ sdk.NewDecCoin(lscosmosKeeper.GetHostChainParams(ctx).MintDenom,
↳ sdk.NewInt(1000000)), cValue)
```

```
95      suite.True(sdk.NewInt64Coin(lscosmosKeeper.GetIBCDenom(ctx),
↳ 1021000).IsEqual(tokenValue))
96      suite.True(sdk.NewDecCoinFromDec(lscosmosKeeper.GetIBCDenom(
↳ ctx), sdk.ZeroDec()).IsEqual(residue))
97
98      cValue = lscosmosKeeper.GetCValue(ctx)
99      stkValue, residue = lscosmosKeeper.ConvertTokenToStk(ctx, sdk.
↳ NewDecCoin(lscosmosKeeper.GetIBCDenom(ctx), sdk.NewInt(1000000)),
↳ cValue)
100     suite.True(sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(
↳ ctx).MintDenom, 979431).IsEqual(stkValue))
101     suite.True(sdk.NewDecCoinFromDec(lscosmosKeeper.
↳ GetHostChainParams(ctx).MintDenom, sdk.NewDecWithPrec
↳ (929480901077000000, 18)).IsEqual(residue))
102 }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to define an invariant on the c value.

Remediation Plan:

SOLVED: The Persistence team fixed the issue. The added invariant checks the c value is between a pre-defined threshold.

FIX Commit ID

3.2 (HAL-02) PROTOCOL FEE DOES NOT HAVE UPPER BOUND - MEDIUM

Description:

With governance proposal, host chain protocol fee amount can be updated. However, It does not have upper bound. Users can lose their tokens when protocol fee is set to higher amount. It is recommended to implement upper bound on the protocol fee.

Code Location:

/x/lscosmos/keeper/proposal.go, Lines 52

Listing 3

```
1     paramsProposal := types.NewHostChainParams(content.ChainID,
↳ content.ConnectionID, content.TransferChannel,
2         content.TransferPort, content.BaseDenom, content.MintDenom
↳ , content.PstakeParams.PstakeFeeAddress,
3         content.MinDeposit, content.PstakeParams.PstakeDepositFee,
↳  content.PstakeParams.PstakeRestakeFee,
4         content.PstakeParams.PstakeUnstakeFee, content.
↳ PstakeParams.PstakeRedemptionFee)
```

Proof Of Concept:

Listing 4

```
1 ./pstaked tx gov submit-proposal pstake-lscosmos-register-host-
↳ chain proposal.json --from alice --fees 1000stake --gas 200000
2
3 {
4   "title": "register host chain proposal",
5   "description": "this proposal register host chain params in the
↳ chain",
6   "module_enabled": true,
7   "chain_id": "test-1",
```

```
  "connection_id": "connection-0",
  "transfer_channel": "channel-0",
  "transfer_port": "transfer",
  "base_denom": "uatom",
  "mint_denom": "ustkatom",
  "min_deposit": "1",
  "allow_listed_validators": {
    "allow_listed_validators": [
      {
        "validator_address": "
↳ cosmosvaloper1hcqg5wj9t42zawqkqucs7la85ffyv08le09ljt",
        "target_weight": "1"
      }
    ]
  },
  "pstake_params": {
    "pstake_deposit_fee": "1.0",
    "pstake_restake_fee": "1.0",
    "pstake_unstake_fee": "1.0",
    "pstake_redemption_fee": "1.0",
    "pstake_fee_address": "
↳ persistence108cqtjz7gqasctvrw74kewg6642062kmfuujsd"
  }
  "deposit": "1000000stake"
}
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

It is recommended to define an upper bound on the protocol fee.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by defining the threshold on the deposit-restake-unstake fee.

FIX Commit ID

3.3 (HAL-03) MISSING FUNCTIONALITY FOR MOVING REMAINING TOKENS TO PSTAKEADDRESS - MEDIUM

Description:

AfterEpochEnd handle multiple epochs for stake, reward and undelegate. DelegationEpochWorkflow is used to generate delegate transaction to delegate amount of stake accumulated over the epoch. However, the remaining balance is not moved to pstake address at the end of function. The function is missing implementation.

Code Location:

/x/lscosmos/keeper/hooks.go#L114

Listing 5

```
1 func (k Keeper) DelegationEpochWorkFlow(ctx sdk.Context,
↳ hostChainParams lscosmostypes.HostChainParams) {
2 ...
3
4     ctx.EventManager().EmitEvents(res.GetEvents())
5
6     // move extra tokens to pstake address - anyone can send
↳ tokens to delegation address.
7     // should be transferred to pstake address.
8     //remainingBalance := allBalances.Sub(sdk.NewCoins(
↳ depositBalance))
9 }
```

Proof Of Concept:

Listing 6

```
func (suite *IntegrationTestSuite) TestAfterEpochEnd() {
    app, ctx := suite.app, suite.ctx

    lscosmosKeeper := app.LSCosmosKeeper

    // enable the module
    lscosmosKeeper.SetModuleState(ctx, true)

    // calling the rewards epoch identifier without setting delegation state
    // to go into len check of Rewards workflow
    suite.Require().NoError(app.LSCosmosKeeper.AfterEpochEnd(ctx, types.RewardEpochIdentifier, 1))

    // get host chain params
    hostChainParams := lscosmosKeeper.GetHostChainParams(ctx)

    // create delegations state for reward epoch
    delegationState := types.DelegationState{
        HostAccountDelegations: []types.HostAccountDelegation{
            {
                ValidatorAddress: "",
                Amount:           sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).BaseDenom, 600000),
            },
            {
                ValidatorAddress: "",
                Amount:           sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).BaseDenom, 200000),
            },
            {
                ValidatorAddress: "",
                Amount:           sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).BaseDenom, 100000),
            },
            {
                ValidatorAddress: "",
                Amount:           sdk.NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).BaseDenom, 100000),
            },
        },
```

```
36          HostDelegationAccountBalance: sdk.NewCoins(sdk.
↳ NewInt64Coin(lscosmosKeeper.GetHostChainParams(ctx).BaseDenom,
↳ 1000)),
37      }
38
39      // set the above created delegation state in module
40      lscosmosKeeper.SetDelegationState(ctx, delegationState)
41
42      // create ibcDenom from host chain params
43      ibcDenom := ibctransfertypes.ParseDenomTrace(
44          ibctransfertypes.GetPrefixedDenom(
45              hostChainParams.TransferPort, hostChainParams.
↳ TransferChannel, hostChainParams.BaseDenom,
46          ),
47      ).IBCDenom()
48
49      // mint coins in module with ibcDenom
50      err := app.BankKeeper.MintCoins(ctx, types.ModuleName, sdk.
↳ NewCoins(sdk.NewCoin(ibcDenom, sdk.NewInt(20000))))
51      suite.NoError(err)
52
53      // send the minted coins to types.DepositModuleAccount
54      err = app.BankKeeper.SendCoinsFromModuleToModule(ctx,
55          types.ModuleName,
56          types.DepositModuleAccount,
57          sdk.NewCoins(sdk.NewCoin(ibcDenom, sdk.NewInt(10000))),
58      )
59      suite.NoError(err)
60
61      // call the after epoch end of LSCosmosKeeper to perform the
↳ actions
62      suite.Require().NoError(app.LSCosmosKeeper.AfterEpochEnd(ctx,
↳ types.DelegationEpochIdentifier, 1))
63      suite.Require().NoError(app.LSCosmosKeeper.AfterEpochEnd(ctx,
↳ types.UndelegationEpochIdentifier, 1))
64 }
65
```

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

Ensue that remaining tokens are moved to pStake address.

Remediation Plan:

SOLVED: The Persistence team solved this issue by changing the **logic of epoch workflow**.

Commit ID: Commit ID

3.4 (HAL-04) NEW HOST CHAIN PROPOSAL PARAMETERS ARE NOT VALIDATED - MEDIUM

Description:

The function **NewRegisterHostChainProposal** creates a new host chain change proposal. However, any of the proposal parameters are not validated. **ValidateBasic** is happening during the **CheckTx** phase, and it doesn't have access to the state. So, it can verify only the current object. For complex objects, it can be an intensive operation. This function can be utilized on the proposal generation.

Code Location:

/x/lscosmos/types/governance_proposal.go, Lines 27

Listing 7

```
1 func (m *RegisterHostChainProposal) ValidateBasic() error {
2     err := govtypes.ValidateAbstract(m)
3     if err != nil {
4         return err
5     }
6
7     return nil
8 }
```

Scenario:

- Register proposal with same base and mint denom.
- Connection will be checked and proposal will be registered.
- However, during the stake operations, same denom could not proceed by the code base.

Proof Of Concept:

Listing 8

```
./pstaked tx gov submit-proposal pstake-lscosmos-register-host-chain proposal.json --from alice --fees 1000stake --gas 200000

{
  "title": "register host chain proposal",
  "description": "this proposal register host chain params in the chain",
  "module_enabled": true,
  "chain_id": "test-1",
  "connection_id": "connection-0",
  "transfer_channel": "channel-0",
  "transfer_port": "transfer",
  "base_denom": "uatom",
  "mint_denom": "ustkatom",
  "min_deposit": "1",
  "allow_listed_validators": {
    "allow_listed_validators": [
      {
        "validator_address": "cosmosvaloper1hcqg5wj9t42zawqkqucs7la85ffyv08le09ljt",
        "target_weight": "1"
      }
    ]
  },
  "pstake_params": {
    "pstake_deposit_fee": "1.0",
    "pstake_restake_fee": "1.0",
    "pstake_unstake_fee": "1.0",
    "pstake_redemption_fee": "1.0",
    "pstake_fee_address": "persistence108cqtjz7gqasctvrw74kewg6642062kmfuujsd"
  }
  "deposit": "1000000stake"
}
```

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

Ensure that all variables are validated.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by adding the fee and denom validation.

FIX Commit ID

3.5 (HAL-05) NON-DETERMINISTIC ITERATIONS CAN CAUSE CONSENSUS FAILURES - MEDIUM

Description:

The codebase contains instances of iteration over maps. As maps are not ordered in Go, iterations over maps are non-deterministic. If the consensus logic expects a deterministic result from the iteration, errors can occur where different nodes reach different states due to the differences in map ordering. If the nodes reach differing states, this can cause consensus issues, which could result in a chain halt.

Code Location:

/x/lscosmos/keeper/proposal.go, Lines 52

Listing 9

```
func DivideUndelegateAmountIntoValidatorSet(sortedValDiff types.
↳ WeightedAddressAmounts, coin sdk.Coin) ([]types.ValAddressAmount,
↳ error) {
    if coin.IsZero() {
        return nil, nil
    }

    // Undelegate first from zero weighted validators then nonzero
↳  weighted
    zeroWeighted, nonZeroWeighted := types.
↳ GetZeroNonZeroWightedAddrAmts(sortedValDiff)
    sort.Sort(sort.Reverse(zeroWeighted))
    sort.Sort(sort.Reverse(nonZeroWeighted))
    valWeighted := append(zeroWeighted, nonZeroWeighted...)

    valAmounts, remainderCoin := distributeCoinsAmongstValSet(
↳ valWeighted, coin)

    // If the remaining amount is not positive, return early
    if !remainderCoin.IsPositive() {
```

```
16          return valAmounts, nil
17      }
18
19      // Divide the remaining amount among the validators a/c to
↳ weight
20      zeroValued := sortedValDiff.GetZeroValued()
21      valAddressMap := types.GetWeightedAddressMap(zeroValued)
22      valAmounts = divideAmountWeightedSet(valAmounts, remainderCoin
↳ , valAddressMap)
23
24      return valAmounts, nil
25  }
```

Proof Of Concept:

Listing 10

```
1  func (suite *IntegrationTestSuite)
↳ TestDivideAmountIntoStateValidatorSet() {
2      _, ctx := suite.app, suite.ctx
3
4      // Test data
5      testMatrix := []struct {
6          state    map[string][]string
7          given    int64
8          expected map[string]int64
9      }{
10         {
11             state: map[string][]string{
12                 "cosmosvalidatorAddr1": {"4000000", "0.1"},
13                 "cosmosvalidatorAddr2": {"8000000", "0.2"},
14                 "cosmosvalidatorAddr3": {"20000000", "0.2"},
15                 "cosmosvalidatorAddr4": {"8000000", "0.5"},
16             },
17             given: 13028679724,
18             expected: map[string]int64{
19                 "cosmosvalidatorAddr1": 1302867972,
20                 "cosmosvalidatorAddr2": 2605735944,
21                 "cosmosvalidatorAddr3": 6514339864,
22                 "cosmosvalidatorAddr4": 2605735944,
23             },
24         },
```

```
25      }
26
27      // Create input parameters
28      for _, test := range testMatrix {
29          // Set validator set weighted amount
30          state := createStateFromMap(test.state, HostStakingDenom)
31          suite.SetupAllowListedValSetAndDelegationState(state)
32
33          // Create state
34          givenCoin := sdk.NewInt64Coin(HostStakingDenom, test.given
↳ )
35          expectedMap := map[string]int64{}
36          for k, v := range test.expected {
37              valAddress, _ := Bech32ifyValAddressBytes(types.
↳ CosmosValOperPrefix, sdk.ValAddress(k))
38              expectedMap[valAddress] = v
39          }
40          allowlistedVals := suite.app.LSCosmosKeeper.
↳ GetAllowListedValidators(ctx)
41          delegationState := suite.app.LSCosmosKeeper.
↳ GetDelegationState(ctx)
42
43          // Run getIdealCurrentDelegations function with params
44
45          valAmounts, err := keeper.FetchValidatorsToDelegate(
↳ allowlistedVals, delegationState, givenCoin)
46          suite.Nil(err, "Error is not nil for validator to delegate
↳ ")
47          // Check outputs
48          actualMap := map[string]int64{}
49          for _, va := range valAmounts {
50              actualMap[va.ValidatorAddr] = va.Amount.Amount.Int64()
51          }
52          suite.Equal(expectedMap, actualMap, "Matching val
↳ distribution")
53      }
54  }
```

Recommendation:

We recommend sorting the map keys into a slice and iterating over the sorted keys to ensure deterministic results among all validators.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by adding sorting on the validator states.

FIX Commit ID

3.6 (HAL-06) ONTIMEOUTPACKET METHOD NOT IMPLEMENTED - LOW

Description:

Operations that involve transferring and writing tokens can get into a state where the chain updates status based on successful transfers, but ignores timeouts. This can lead to balances getting out of sync across chains and eventually users may not be able to redeem their tokens successfully or may be able to redeem more than they should.

Code Location:

/x/lscosmos/keeper/hooks.go#L309

Listing 11

```
1 func (k Keeper) OnTimeoutIBCTransferPacket(ctx sdk.Context, packet
↳  channeltypes.Packet, relayer sdk.AccAddress, transferTimeoutErr
↳ error) error {
2     // Do nothing because amount will be reverted to
↳ delegationModuleAccount.
3     return nil
4 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

We recommend implementing custom logic that handles different possible packet timeouts in order to ensure that the state is correctly synced cross-chain. As an example, the transfer module implements this method, see here.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by adding timeout package functionality.

FIX Commit ID

3.7 (HAL-07) MUSTACCADDRESSFROMBECH32 CAN LEADS TO PANIC IN THE CHAIN - LOW

Description:

AfterEpochEnd function handles the **stake**, **reward** and **undelegate** epoch. During the code review, It has been observed that **PstakeFeeAddress** is checked with **MustAccAddressFromBech32** function. According to SDK documentation, **MustAccAddressFromBech32** calls **AccAddressFromBech32** and panics on error. Even if **PstakeFeeAddress** is multiple times validated, incorrect address can lead to panic in the chain.

Code Location:

/x/lscosmos/keeper/hooks.go#L150

Listing 12

```
    if !remainingDelegationBalance.Empty() {
        feeAddr := sdk.MustAccAddressFromBech32(hostChainParams.
↳ PstakeParams.PstakeFeeAddress)
        err := k.bankKeeper.SendCoinsFromModuleToAccount(ctx,
↳ lscosmostypes.DelegationModuleAccount, feeAddr,
↳ remainingDelegationBalance)
        if err != nil {
            k.Logger(ctx).Error(fmt.Sprintf("could not send
↳ remaining balance: %s in delegationModuleAccount: %s with error: %
↳ s", remainingDelegationBalance, lscosmostypes.
↳ DelegationModuleAccount, err))
            return err
        }
    }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to change **MustAccAddressFromBech32** with **AccAddress-FromBech32** and check return value of an address validation.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by changing implementation with **AccAddressFromBech32**.

FIX Commit ID

3.8 (HAL-08) FEE ADDRESS IS DEFINED AS AN ACCOUNT - LOW

Description:

During the code review, It has been noticed that fee receiver address is inherited from the **AccAddress**. **AccAddress** identifies users (the sender of a message) If the private key stolen by **PstakeFeeAddress** address, all collected fees will be lost by the system.

Code Location:

/x/lscosmos/keeper/msg_server.go#L113

Listing 13

```
    if protocolCoin.IsPositive() {
        err = m.SendProtocolFee(ctx, sdktypes.NewCoins(protocolCoin), types.ModuleName, hostChainParams.PstakeParams.PstakeFeeAddress)
        if err != nil {
            return nil, sdkerrors.Wrapf(
                types.ErrFailedDeposit, "failed to send protocol fee to pstake fee address %s, got error : %s",
                hostChainParams.PstakeParams.PstakeFeeAddress, err,
            )
        }
    }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to change fee address with module.

Remediation Plan:

RISK ACCEPTED: The Persistence team accepted the risk of issue. The Team claims that the fee address will keep treasury funds. From that reason, It is defined as an account.

3.9 (HAL-09) LACK OF SIMULATION AND FUZZING OF THE MODULE INVARIANT - LOW

Description:

The Persistence system lacks comprehensive CosmosSDK simulations and invariants for its **x/lscosmos** module. More thorough use of the simulation feature would facilitate fuzz testing of the entire blockchain and help ensure that the invariants hold.

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

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

- All module 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.
- 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 Persistence team accepted the risk of issue. The team will add simulation/fuzzing scripts after launch.

3.10 (HAL-10) LACK OF SPEC ON THE MODULE - LOW

Description:

The spec file intends to outline the common structure for specifications within this directory. In the pStake's **lscosmos** module is missing spec. This documentation is segmented into developer-focused messages and end-user-facing messages. These messages may be shown to the end user (the human) at the time that they will interact with the module.

Risk Level:

Likelihood - 1
Impact - 3

Code Location:

/x/lscosmos/spec

Recommendation:

It is recommended that modules are fully annotated using spec for all available functionalities.

Remediation Plan:

RISK ACCEPTED: The Persistence team accepted the risk of issue. The team will add specs after launch.

3.11 (HAL-11) COMPUTATIONALLY HEAVY OPERATIONS IN BEGINBLOCKER MAY SLOW DOWN OR STOP BLOCK PRODUCTION - LOW

Description:

BeginBlocker and **EndBlocker** are a way for module developers to add automatic execution of logic to their module. This is a powerful tool that should be used carefully, as complex automatic functions can slow down or even halt the chain. There is a one module within the scope of this audit where the **BeginBlocker** or **EndBlocker** contains unbounded loops that can slow or even halt the chain.

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

We recommend reworking the **BeginBlocker** and **Endblocker** functions in order to reduce their computational complexity.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by calculating the complexity on the **BeginBlocker** and **EndBlocker**.

FIX Commit ID

3.12 (HAL-12) TEST DOCKER IMAGE RUNNING AS ROOT - LOW

Description:

Docker containers usually run with root privileges by default. This allows for unrestricted container management, which means a user could install system packages, edit configuration files, bind privileged ports, etc. During the static analysis, it has been observed that docker image is maintained via root user.

Code Location:

Dockerfile

Listing 14

```
 1 FROM golang:1.18-alpine AS build-env
 2
 3 # Set up dependencies
 4 ENV PACKAGES curl make git libc-dev bash gcc linux-headers eudev-
↳ dev python3
 5
 6 # Set working directory for the build
 7 WORKDIR /go/src/github.com/persistenceOne/pstake-native
 8
 9 # Add source files
10 COPY . .
11
12 # Install minimum necessary dependencies, build Cosmos SDK, remove
↳  packages
13 RUN apk add --no-cache $PACKAGES && \
14   make install
15
16 # Final image
17 FROM alpine:edge
18
19 # Install ca-certificates
20 RUN apk add --update ca-certificates jq bash curl
21 WORKDIR /root
```

```
22
23 # Copy over binaries from the build-env
24 COPY --from=build-env /go/bin/pstaked /usr/bin/pstaked
25
26 EXPOSE 26657
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to build Dockerfile and run container as a non-root user.

Listing 15: Reference

```
1 USER 1001: this is a non-root user UID, and here it is assigned to
↳  the image to run the current container as an unprivileged user.
↳ By doing so, the added security and other restrictions mentioned
↳ above are applied to the container.
```

Remediation Plan:

RISK ACCEPTED: The Persistence team accepted the risk of issue.

3.13 (HAL-13) DELEGATE MESSAGES WITH ZERO AMOUNT IS NOT CHECKED - LOW

Description:

DelegateMsgs function gives the list of **Delegate Txs** to be executed based on the current state and parameters. However, the amount is not checked If It is bigger than zero. During the IBC message execution, zero amount of delegations can fail on the system.

Code Location:

/x/lscosmos/keeper/validator_strategy.go#L13-L39

Listing 16

```
func (k Keeper) DelegateMsgs(ctx sdk.Context, delegatorAddr string
↳ , amount sdk.Int, denom string) ([]sdk.Msg, error) {
    valList := k.GetAllowListedValidators(ctx)
    delegationState := k.GetDelegationState(ctx)

    valAddressAmount, err := FetchValidatorsToDelegate(valList,
↳ delegationState, sdk.NewCoin(denom, amount))
    if err != nil {
        return nil, err
    }

    msgs := make([]sdk.Msg, len(valAddressAmount))

    for i, val := range valAddressAmount {

        msg := &stakingtypes.MsgDelegate{
            DelegatorAddress: delegatorAddr,
            ValidatorAddress: val.ValidatorAddr,
            Amount:           val.Amount,
        }
        msgs[i] = msg
    }

    return msgs, nil
```

```
23 }
24
25 // UndelegateMsgs gives the list of Undelegate Txs to be executed
↳ based on the current state and params.
26 func (k Keeper) UndelegateMsgs(ctx sdk.Context, delegatorAddr
↳ string, amount sdk.Int, denom string) ([]sdk.Msg, []types.
↳ UndelegationEntry, error) {
27     valList := k.GetAllowListedValidators(ctx)
28     delegationState := k.GetDelegationState(ctx)
29
30     valAddressAmount, err := FetchValidatorsToUndelegate(valList,
↳ delegationState, sdk.NewCoin(denom, amount))
31     if err != nil {
32         return nil, nil, err
33     }
34
35     msgs := make([]sdk.Msg, len(valAddressAmount))
36     undelegationEntries := make([]types.UndelegationEntry, len(
↳ valAddressAmount))
37 ...
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to check the amount is bigger than zero.

Remediation Plan:

SOLVED: The Persistence team fixed the adding amount check.

FIX Commit ID

3.14 (HAL-14) GENESIS STATE DOES NOT HAVE ENOUGH VALIDATION - LOW

Description:

The InitGenesis method is executed during InitChain when the application is started. Given a GenesisState, it initializes the subset of the state managed by the module by using the module's keeper setter function on each parameter within the GenesisState. During the chain initialization, module parameters can be validated again with its parameters.

Code Location:

genesis.go

Listing 17

```
func InitGenesis(ctx sdk.Context, k keeper.Keeper, genState types.GenesisState) {
    // this line is used by starport scaffolding # genesis/module/init

    k.SetParams(ctx, genState.Params)
    k.SetModuleState(ctx, genState.ModuleEnabled)
    k.SetHostChainParams(ctx, genState.HostChainParams)
    if !genState.HostChainParams.IsEmpty() {
        err := k.NewCapability(ctx, host.ChannelCapabilityPath(genState.HostChainParams.TransferPort, genState.HostChainParams.TransferChannel))
        if err != nil {
            panic(err)
        }
    }
    k.SetAllowListedValidators(ctx, genState.AllowListedValidators)
    k.SetDelegationState(ctx, genState.DelegationState)
    k.SetHostChainRewardAddress(ctx, genState.HostChainRewardAddress)
    k.SetIBCTransientStore(ctx, genState.IBCAmountTransientStore)

```

```
18      //TODO add all stores here and in export.
19
20      k.GetDepositModuleAccount(ctx)
21      k.GetDelegationModuleAccount(ctx)
22      k.GetRewardModuleAccount(ctx)
23      k.GetUndelegationModuleAccount(ctx)
24
25  }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Add necessity validation mechanisms on the genesis.

Remediation Plan:

SOLVED: The Persistence team fixed the adding validation on the genesis state.

FIX Commit ID

3.15 (HAL-15) CODECS DO NOT HAVE INIT FUNCTION - LOW

Description:

Codec **init** function registers all Amino interfaces and concrete types on the module's amino codec. On the code base, **init** function is not implemented.

Code Location:

/x/lscosmos/types/codec.go#L43

Listing 18

```
var (
    Amino     = codec.NewLegacyAmino()
    ModuleCdc = codec.NewProtoCodec(cdctypes.NewInterfaceRegistry
↳ ())
)
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Amino types should be ideally registered inside this codec within the **init** function of each module's **codec.go**.

Remediation Plan:

SOLVED: The Persistence team fixed the adding init function.

FIX Commit ID

3.16 (HAL-16) USE REGISTERLEGACYAMINOCODEC INSTEAD OF REGISTERCODE - LOW

Description:

RegisterLegacyAminoCodec registers the **lscosmos** module's types for the given codec. However, It's still using **RegisterCodec** instead of **RegisterLegacyAminoCodec**.

Code Location:

/x/lscosmos/module.go#L51

Listing 19

```
1 func (AppModuleBasic) RegisterCodec(cdc *codec.LegacyAmino) {
2     types.RegisterCodec(cdc)
3 }
4
5 func (AppModuleBasic) RegisterLegacyAminoCodec(cdc *codec.
↳ LegacyAmino) {
6     types.RegisterCodec(cdc)
7 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider using **RegisterLegacyAminoCodec**.

Remediation Plan:

SOLVED: The Persistence team fixed the registering the correct codec.

FIX Commit ID

3.17 (HAL-17) CROSS SITE SCRIPTING THROUGH GOVERNANCE PROPOSAL - LOW

Description:

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. During host chain proposal registration, **Title** and **Description** parameters are not sanitized. If the user interface is directly parsing these values, Cross site scripting vulnerability can be observed on the user interface.

Code Location:

/x/lscosmos/client/cli/tx.go#L124

Listing 20

```
            content := types.NewRegisterHostChainProposal(
                proposal.Title,
                proposal.Description,
                proposal.ModuleEnabled,
                proposal.ChainID,
                proposal.ConnectionID,
                proposal.TransferChannel,
                proposal.TransferPort,
                proposal.BaseDenom,
                proposal.MintDenom,
                proposal.PstakeParams.PstakeFeeAddress,
                minDeposit,
                proposal.AllowListedValidators,
                depositFee,
                restakeFee,
                unstakeFee,
                redemptionFee,
            )
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to sanitize user input. By sanitizing means, every payload goes through the filtering process. Bluemonday can be utilized for this purpose.

Remediation Plan:

RISK ACCEPTED: The Persistence team accepted the risk of issue.

3.18 (HAL-18) TYPO ON THE VARIABLES - INFORMATIONAL

Description:

During the code review, It has been observed that baseDenomUndelegtions and AttributeAmountRecieved have typo.

Code Location:

/x/lscosmos/keeper/msg_server.go#L255

Listing 21

```
    baseDenomUndelegtions, _ := m.ConvertStkToToken(ctx, sdktypes.
↳ NewDecCoinFromCoin(undelegations.TotalUndelegationAmount), m.
↳ GetCValue(ctx))

```

/x/lscosmos/keeper/msg_server.go#L126

Listing 22

```
    ctx.EventManager().EmitEvents(sdktypes.Events{
        sdktypes.NewEvent(
            types.EventTypeLiquidStake,
            sdktypes.NewAttribute(types.AttributeDelegatorAddress,
↳  delegatorAddress.String()),
            sdktypes.NewAttribute(types.AttributeAmount, mintToken
↳ .String()),
            sdktypes.NewAttribute(types.AttributeAmountRecieved,
↳ mintToken.Sub(protocolCoin).String()),
            sdktypes.NewAttribute(types.AttributePstakeDepositFee,
↳  protocolCoin.String()),
        ),
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider changing the baseDenomUndelegtions with baseDenomUndelegations and AttributeAmountRecieved with AttributeAmountReceieved.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by changing variable names.

FIX Commit ID

3.19 (HAL-19) OPEN TODOs - INFORMATIONAL

Description:

Open To-dos can point to architecture or programming issues that still need to be resolved. Often these kinds of comments indicate areas of complexity or confusion for developers. This provides value and insight to an attacker who aims to cause damage to the protocol.

Code Location:

TO-DO:

Listing 23: Open Todos

```
./x/lscosmos/types/keys.go:63:  UndelegationCompletionTimeBuffer
↳       = time.Second * 10 //TODO change
./x/lscosmos/keeper/handshake.go:94:    //TODO more checks,
↳ capability, channelID??
./x/lscosmos/keeper/handshake.go:178:   // TODO add checks for
↳ capabilities, ports, channels
./x/lscosmos/keeper/handshake.go:202:   eventType := "lscosmos-ack
↳ " //TODO rename
./x/lscosmos/keeper/handshake.go:207:       // TODO: handle for
↳ sdk 0.46.x
./x/lscosmos/keeper/handshake.go:303:   // TODO add checks for
↳ capabilities, ports, channels
./x/lscosmos/keeper/handshake.go:317:       // TODO: handle for
↳ sdk 0.46.x
./x/lscosmos/keeper/handshake.go:459:           //TODO disable
↳ module?
./x/lscosmos/keeper/proposal.go:40: // TODO Understand
↳ capabilities and see if it has to be/ should be claimed in
↳ lsscopedkeeper. If it even matters.
./x/lscosmos/keeper/proposal_test.go:3:// todo see how to create
↳ channel in tests
./x/lscosmos/genesis.go:29: //TODO add all stores here and in
↳ export.
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider resolving the To-dos before deploying code to a production context. Use an independent issue tracker or other project management software to track development tasks.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by resolving to-dos.

FIX Commit ID

3.20 (HAL-20) PANIC IS USED FOR ERROR HANDLING - INFORMATIONAL

Description:

Several instances of the panic function were identified in the codebase. They appear to be used to handle errors. This can cause potential issues, as invoking a panic can cause the program to halt execution and crash in some cases. This in turn can negatively impact the availability of the software for users.

Code Location:

Listing 24: Instances of panic identified in the codebase

```
./x/lscosmos/types/msgs.go:57:      panic(err)
./x/lscosmos/types/msgs.go:103:     panic(err)
./x/lscosmos/types/msgs.go:145:     panic(err)
./x/lscosmos/types/msgs.go:187:     panic(err)
./x/lscosmos/types/msgs.go:225:     panic(err)
./x/lscosmos/keeper/delegation_state.go:83:
↳ delegationState.HostAccountDelegations[i].Amount =
↳ existingDelegation.Amount.Sub(delegation.Amount) //This will panic
↳  if coin goes negative
./x/lscosmos/keeper/delegation_state.go:126:        panic("Adding
↳ Unbonding entries for non existing epoch")
./x/lscosmos/keeper/unbonding_epoch_entries.go:13:      panic(err)
./x/lscosmos/genesis.go:21:         panic(err)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Instead of using panics, custom errors should be defined and handled according to the Best practices.

Remediation Plan:

ACKNOWLEDGED: The Persistence team acknowledged the issue.

3.21 (HAL-21) RENAME TO THE MODULE - INFORMATIONAL

Description:

The **validator_strategy** codebase generate delegate and undelegate messages with the current system state and parameters. However, the name of the file can cause misunderstandings.

Code Location:

/x/lscosmos/keeper/validator_strategy.go

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider changing filename with **delegation_strategy**.

Remediation Plan:

SOLVED: The Persistence team fixed the issue by changing the source code filename.

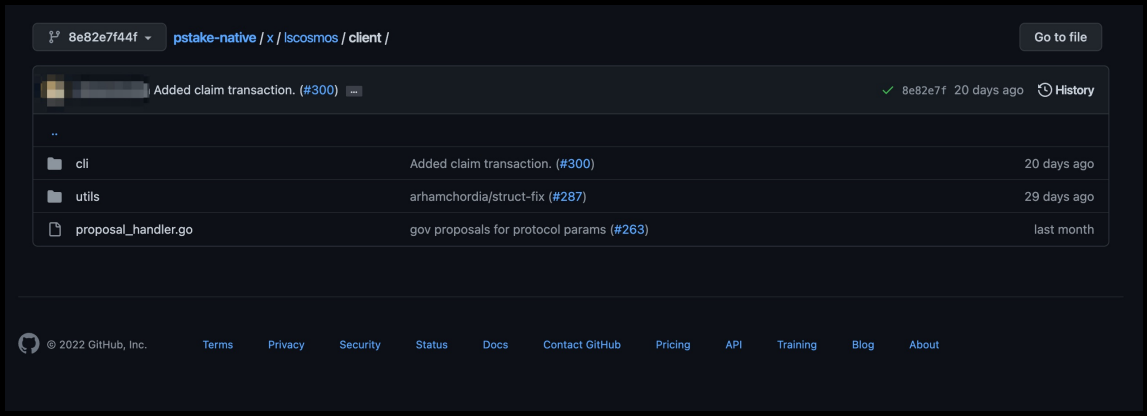
FIX Commit ID

3.22 (HAL-22) LSCOSMOS DOES NOT USE REST CLI HANDLER - INFORMATIONAL

Description:

During the code review, It has been observed that **CosmosSDK** REST handler is not used in the module.

Location:



Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Evaluate whether the CosmosSDK REST interface is needed by the module. This package provides HTTP types and primitives for REST requests validation and responses handling.

Remediation Plan:

ACKNOWLEDGED: The Persistence team acknowledged the issue.

3.23 (HAL-23) REGISTERGRPCGATEWAYROUTES IS NOT FUNCTIONAL - INFORMATIONAL

Description:

RegisterGRPCGatewayRoutes registers the **gRPC** Gateway routes for the module, and The gRPC-Gateway allows you to add custom routes. However, in the module, grpc gateway routes are not registered.

Risk Level:

Likelihood - 1
Impact - 1

Code Location:

/x/lscosmos/module.go#L78

Listing 25

```
1 // RegisterGRPCGatewayRoutes registers the gRPC Gateway routes for
↳  the module.
2 func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.
↳ Context, mux *runtime.ServeMux) {
3     // this line is used by starport scaffolding # 2
4 }
```

Recommendation:

Consider registering the GRPC routes.

Remediation Plan:

ACKNOWLEDGED: The Persistence team acknowledged the issue.

AUTOMATED TESTING

Description:

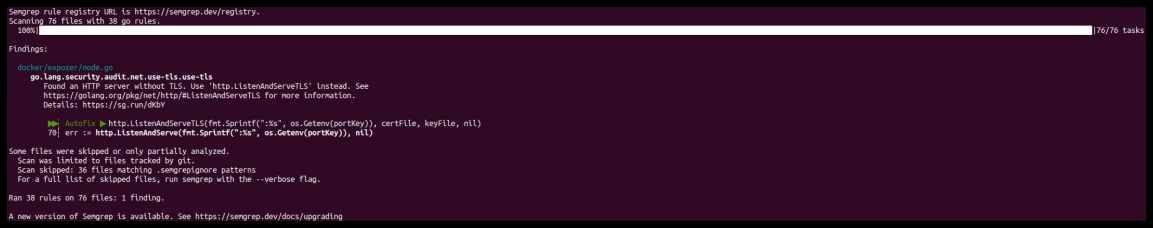
Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec, semgrep, unconvert, LGTM and Nancy. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, 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:



Golang Linter - Security Analysis Output:

Listing 27

```
WARN [runner] The linter 'golint' is deprecated (since v1.41.0) due to: The repository of the linter has been archived by the owner.  Replaced by revive.
WARN [runner] The linter 'maligned' is deprecated (since v1.38.0) due to: The repository of the linter has been archived by the owner.  Replaced by govet 'fieldalignment'.
WARN [runner] The linter 'interfacer' is deprecated (since v1.38.0) due to: The repository of the linter has been archived by the owner.
WARN [runner] The linter 'scopelint' is deprecated (since v1.39.0) due to: The repository of the linter has been deprecated by the owner.  Replaced by exportloopref.
WARN [runner] The linter 'structcheck' is deprecated (since v1.49.0) due to: The owner seems to have abandoned the linter.  Replaced by unused.
WARN [runner] The linter 'deadcode' is deprecated (since v1.49.0) due to: The owner seems to have abandoned the linter.  Replaced by unused.
WARN [linters_context] structcheck is disabled because of generics. You can track the evolution of the generics support by following the https://github.com/golangci/golangci-lint/issues/2649.
WARN [runner/nolint] Found unknown linters in //nolint directives: len_not_fixed, nocredentials, skip_for_now, unused_vars, used_for_swagger_ui_docs
docker/exposer/node.go:70:9: G114: Use of net/http serve function that has no support for setting timeouts (gosec)
    err := http.ListenAndServe(fmt.Sprintf(":%s", os.Getenv(portKey)), nil)

```

Nancy - Security Analysis Output Sample:

```
pkg:golang/github.com/tendermint/tendermint@v0.34.20
1 known vulnerabilities affecting installed version
```

1 vulnerability found	
Description	1 non-CVE vulnerability found. To see more details, please create a free account at https://ossindex.sonatype.org/ and request for this information using your registered account
OSS Index ID	sonatype-2021-0598
CVSS Score	4.8/10 (Medium)
CVSS Vector	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:N
Link for more info	https://ossindex.sonatype.org/vulnerability/sonatype-2021-0598

```
pkg:golang/golang.org/x/text@v0.3.7
1 known vulnerabilities affecting installed version
```

[CVE-2022-32149] CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')	
Description	An attacker may cause a denial of service by crafting an Accept-Language header which ParseAcceptLanguage will take significant time to parse.
OSS Index ID	CVE-2022-32149
CVSS Score	7.5/10 (High)
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Link for more info	https://ossindex.sonatype.org/vulnerability/CVE-2022-32149?component-type=golang&component-name=golang.org%2Fx%2Ftext&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev

```
pkg:golang/nhooyr.io/websocket@v1.8.6
1 known vulnerabilities affecting installed version
```

1 vulnerability found	
Description	1 non-CVE vulnerability found. To see more details, please create a free account at https://ossindex.sonatype.org/ and request for this information using your registered account
OSS Index ID	sonatype-2021-0456
CVSS Score	7.5/10 (High)
CVSS Vector	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Link for more info	https://ossindex.sonatype.org/vulnerability/sonatype-2021-0456

```
3 Vulnerable Packages
```

Summary	
Audited Dependencies	97
Vulnerable Dependencies	3

THANK YOU FOR CHOOSING

// HALBORN