Elys Modules

Elys Network

HALBORN

Elys Modules - Elys Network

Prepared by: HALBORN

Last Updated 04/09/2024

Date of Engagement: February 2nd, 2024 - March 15th, 2024

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
22	2	2	3	8	7

TABLE OF CONTENTS

1. Introduction

`The Elys Network Team` engaged Halborn to conduct a security assessment on their cosmos appchain modules, beginning on *02/02/2024* and ending on *03/15/2024*. The security assessment was scoped to the sections of code that pertain to the modules of app chain. Commit hashes and further details can be found in the Scope section of this report.

2. Assessment Summary

Halborn was provided 12 weeks for the engagement and assigned 1 full-time security engineer to review the security of the modules in scope. The engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Ensure that the **AppChain Modules** operate as intended.

- Identify potential security issues with the custom modules used in the Elys Chain.

In summary, Halborn identified some security issues that were mostly addressed by the `Elys Network team`.

3. Test Approach And 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 custom modules. 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 assessment:

- Research into architecture and purpose.

- Static Analysis of security for scoped repository, and imported functions. (e.g., `staticcheck`, `gosec`, `unconvert`, `codeql`, `ineffassign` and `semgrep`)

- Manual Assessment for discovering security vulnerabilities on the codebase.

- Ensuring the correctness of the codebase.

- Dynamic Analysis of files and modules related to the **Elys Network Modules**.

3.1 Out-Of-Scope

- External libraries and financial-related attacks.
- New features/implementations after/with the **remediation commit IDs**
- Changes that occur outside of the scope of PRs.

4. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

ATTACK COMPLEXITY (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

METRICS:

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

INTEGRITY (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

AVAILABILITY (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

DEPOSIT (D):

Measures the impact to the deposits made to the contract by either users or owners.

YIELD (Y):

Measures the impact to the yield generated by the contract for either users or owners.

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

SCOPE (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9

SEVERITY	SCORE VALUE RANGE
Low	2 - 4.4
Informational	0 - 1.9

5. SCOPE

URLS

(a) URL:

(b) Items in scope:

- parameter
- assetprofile
- transferhook
- commitment
- stablestake
- oracle
- accountedpool
- amm

Out-of-Scope: Third party dependencies., Economical/Financial Attacks

(a) URL:

(b) Items in scope:

- incentive
- leaverageLP
- tokenomics
- perpetual
- epoch

Out-of-Scope: Third party dependencies., Economical/Financial Attacks

REMEDIATION COMMIT ID:

- https://github.com/elys-network/elys/pull/337/commits/3dfa88b84a7b8f3996ce7f61f9bf2ccfb10d39e1
- https://github.com/elys-network/elys/pull/375/commits/c148ea189cee549e3c9aa61f1c5f7a7defbcbbcd
- https://github.com/elys-network/elys/pull/447/commits/d5d59a802a7d19f94a33cf91bbf77f420528ab25
- https://github.com/elys-network/elys/pull/362/commits/76ac10f8a95f689d62abcf8cad452f8b457f5599
- https://github.com/elys-network/elys/pull/337/commits/537ccb4b7244760b8e774ea509e8d09b124b8514
- https://github.com/elys-network/elys/pull/394/commits/c782c0bd2141a5e805a14ffa25a25adf97be8dbc
- https://github.com/elys-network/elys/pull/447/commits/743916095819033ee6dcb476398fbf2df279a6e8

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
2	2	3	8	7

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
LACK OF ACCESS CONTROL IN REQUESTBANDPRICE FUNCTION	CRITICAL	SOLVED - 03/12/2024
INCORRECT CALCULATION OF MINIMUM COLLATERAL DUE TO IGNORED DECIMAL PRECISION	CRITICAL	SOLVED - 03/12/2024
ABSENCE OF CREATION FEE IN CREATEPOOL METHOD LEADING TO SPAMMING ATTACKS	HIGH	SOLVED - 04/05/2024
INADEQUATE HANDLING OF LARGE WEBASSEMBLY (WASM) FILES ON THE SYSTEM LEADS TO DOS	HIGH	SOLVED - 03/12/2024
ABSENCE OF IBC CHANNEL VERIFICATION IN UPDATEENTRY FUNCTION	MEDIUM	SOLVED - 03/12/2024
LACK OF BROKER ADDRESS VALIDATION IN MSGUPDATEBROKERADDRESS'S VALIDATEBASIC FUNCTION	MEDIUM	SOLVED - 03/12/2024
IMPLEMENT TWAP FOR MORE ACCURATE PRICE RETRIEVAL IN KEEPER METHODS	MEDIUM	RISK ACCEPTED
JOINPOOLNOSWAP EXHIBITS DISCREPANCY BETWEEN EXPECTED AND ACTUAL CALCULATED SHARE OUTPUTS	LOW	RISK ACCEPTED
BULK COIN SENDS	LOW	RISK ACCEPTED
LACK OF SPEC ON THE MODULES	LOW	RISK ACCEPTED
MISSING ASSET WHITELISTING/DENOM CHECK IN FEEDPRICE METHOD	LOW	RISK ACCEPTED

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
INSUFFICIENT VALIDATION IN MSGCREATEENTRY, MSGUPDATEENTRY, AND MSGDELETEENTRY	LOW	RISK ACCEPTED
VULNERABILITIES IN COSMOS SDK V0.47.4 AFFECTING ELYS NETWORK	LOW	RISK ACCEPTED
UTILIZE BLOCK HEIGHT IN PRICE FEEDING	LOW	SOLVED - 04/08/2024
POTENTIAL DENIAL OF SERVICE (DOS) VULNERABILITY IN POWAPPROX FUNCTION	LOW	RISK ACCEPTED
INCREASE PRECISION OF INITIAL POOL SHARES TO MATCH INDUSTRY STANDARDS	INFORMATIONAL	SOLVED - 04/08/2024
MISSING USAGE DESCRIPTION FOR ALL TRANSACTION COMMANDS CLI	INFORMATIONAL	ACKNOWLEDGED
IMPLEMENT FEE MARKET INTEGRATED INTO CONSENSUS LAYER	INFORMATIONAL	ACKNOWLEDGED
LACK OF IBC RATE-LIMITTING IMPLEMENTATION	INFORMATIONAL	ACKNOWLEDGED
ARBITRARY TOKEN TRANSFER LEADS TO CHAIN HALT	INFORMATIONAL	ACKNOWLEDGED
KEEP SLIPPAGE EXPIRATION DATE AS A GOVERNANCE PARAMETER	INFORMATIONAL	SOLVED - 04/08/2024
SILENT ERROR IN RECORDWITHDRAWVALIDATORCOMMISSION FUNCTION	INFORMATIONAL	ACKNOWLEDGED

7. FINDINGS & TECH DETAILS

7.1 LACK OF ACCESS CONTROL IN REQUESTBANDPRICE FUNCTION

// CRITICAL

Description

This function is intended to request price data from the Band Oracle, but it appears to lack sufficient access control measures. This oversight could potentially allow unauthorized entities to manipulate oracle requests, leading to inaccurate or compromised data being fed into the system. The function **RequestBandPrice** does not implement checks to ensure that the caller or the message sender has the necessary permissions to request data from the Band Oracle. In its current state, any entity can potentially trigger the oracle request, which could lead to a range of issues, including but not limited to: Data manipulation Spamming the oracle with unnecessary requests Triggering unintended oracle responses.

```
func (k msgServer) RequestBandPrice(goCtx context.Context, msg *types.MsgRequestBandPrice)
(*types.MsgRequestBandPriceResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    sourcePort := types.PortID
    channelCap, ok := k.scopedKeeper.GetCapability(ctx, host.ChannelCapabilityPath(sourcePort, msg.SourceChannel))
    if !ok {
        return nil, errorsmod.Wrap(channeltypes.ErrChannelCapabilityNotFound,
            "module does not own channel capability")
    }

    encodedCalldata := obi.MustEncode(*msg.Calldata)
    packetData := packet.NewOracleRequestPacketData(
        msg.ClientID,
        msg.OracleScriptID,
        encodedCalldata,
        msg.AskCount,
        msg.MinCount,
        msg.FeeLimit,
        msg.PrepareGas,
        msg.ExecuteGas,
    )

    _, err := k.channelKeeper.SendPacket(ctx, channelCap, sourcePort, msg.SourceChannel, clienttypes.NewHeight(0, 0),
uint64(ctx.BlockTime().UnixNano()+int64(10*time.Minute)), packetData.GetBytes())
    if err != nil {
        return nil, err
    }

    return &types.MsgRequestBandPriceResponse{}, nil
}
```

Proof of Concept

```
func CmdRequestBandPrice() *cobra.Command {
    cmd := &cobra.Command{
        Use:   "request-band-price [oracle-script-id] [requested-validator-count] [sufficient-validator-count]",
        Short: "Make a new BandPrice query request via an existing BandChain oracle script",
        Args:  cobra.ExactArgs(3),
        RunE: func(cmd *cobra.Command, args []string) error {
            // retrieve the oracle script id.
            uint64OracleScriptID, err := strconv.ParseUint(args[0], 10, 64)
            if err != nil {
                return err
            }
            oracleScriptID := types.OracleScriptID(uint64OracleScriptID)

            // retrieve the requested validator count.
            askCount, err := strconv.ParseUint(args[1], 10, 64)
            if err != nil {
                return err
            }

            // retrieve the sufficient(minimum) validator count.
            minCount, err := strconv.ParseUint(args[2], 10, 64)
```

```
            if err != nil {
                return err
            }

            channel, err := cmd.Flags().GetString(flagChannel)
            if err != nil {
                return err
            }

            // retrieve the list of symbols for the requested oracle script.
            symbols, err := cmd.Flags().GetStringSlice(flagSymbols)
            if err != nil {
                return err
            }

            // retrieve the multiplier for the symbols' price.
            multiplier, err := cmd.Flags().GetUint64(flagMultiplier)
            if err != nil {
                return err
            }

            calldata := &types.BandPriceCallData{
                Symbols:    symbols,
                Multiplier: multiplier,
            }

            // retrieve the amount of coins allowed to be paid for oracle request fee from the pool account.
            coinStr, err := cmd.Flags().GetString(flagFeeLimit)
            if err != nil {
                return err
            }
            feeLimit, err := sdk.ParseCoinsNormalized(coinStr)
            if err != nil {
                return err
            }

            // retrieve the amount of gas allowed for the prepare step of the oracle script.
            prepareGas, err := cmd.Flags().GetUint64(flagPrepareGas)
            if err != nil {
                return err
            }

            // retrieve the amount of gas allowed for the execute step of the oracle script.
            executeGas, err := cmd.Flags().GetUint64(flagExecuteGas)
            if err != nil {
                return err
            }

            clientCtx, err := client.GetClientTxContext(cmd)
            if err != nil {
                return err
            }

            msg := types.NewMsgRequestBandPrice(
                clientCtx.GetFromAddress().String(),
                oracleScriptID,
                channel,
                calldata,
                askCount,
                minCount,
                feeLimit,
                prepareGas,
                executeGas,
            )
            if err := msg.ValidateBasic(); err != nil {
                return err
            }
            return tx.GenerateOrBroadcastTxCLI(clientCtx, cmd.Flags(), msg)
        },
    }

    cmd.Flags().String(flagChannel, "", "The channel id")
    cmd.MarkFlagRequired(flagChannel)
    cmd.Flags().StringSlice(flagSymbols, nil, "Symbols used in calling the oracle script")
    cmd.Flags().Uint64(flagMultiplier, 1000000, "Multiplier used in calling the oracle script")
    cmd.Flags().String(flagFeeLimit, "", "the maximum tokens that will be paid to all data source providers")
    cmd.Flags().Uint64(flagPrepareGas, 200000, "Prepare gas used in fee counting for prepare request")
    cmd.Flags().Uint64(flagExecuteGas, 200000, "Execute gas used in fee counting for execute request")
    flags.AddTxFlagsToCmd(cmd)

    return cmd
}
```

BVSS

AO:A/AC:L/AX:L/C:C/I:C/A:N/D:N/Y:N/R:N/S:C (10.0)

Recommendation

Introduce checks at the beginning of the RequestBandPrice function to ensure that only authorized entities can make requests to the Band Oracle.

Remediation Plan

SOLVED: The **Elys Network team** solved this issue by deleting the implementation.

Remediation Hash

https://github.com/elys-network/elys/pull/337/commits/3dfa88b84a7b8f3996ce7f61f9bf2ccfb10d39e1

7.2 INCORRECT CALCULATION OF MINIMUM COLLATERAL DUE TO IGNORED DECIMAL PRECISION

// CRITICAL

Description

The function **CalcMinCollateral** in the keeper package is designed to calculate the minimum collateral required to open a position based on the provided leverage. However, the current implementation does not correctly account for the decimal precision of the collateral amount. The calculation returns the result without considering the **collateral_decimals**, leading to potential inaccuracies in the minimum collateral required, especially in systems where collateral assets have significant decimal precision.

```
// CalcMinCollateral calculates the minimum collateral required to open a position
func (k Keeper) CalcMinCollateral(ctx sdk.Context, leverage sdk.Dec) (math.Int, error) {
    // leverage must be greater than 1
    if leverage.LTE(sdk.NewDec(1)) {
        return sdk.ZeroInt(), errorsmod.Wrapf(types.ErrInvalidLeverage, "leverage must be greater than 1")
    }

    // get min borrow rate
    borrowInterestRateMin := k.GetBorrowInterestRateMin(ctx)

    // min_collateral = ( 1 / (( leverage - 1 ) * borrow_interest_rate_min )) / 10 ^ collateral_decimals
    minCollateral := sdk.NewDec(1).Quo(
        leverage.Sub(sdk.NewDec(1)).Mul(borrowInterestRateMin),
    )

    return minCollateral.TruncateInt(), nil
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:C/A:C/D:N/Y:N/R:N/S:C (10.0)

Recommendation

Users may be required to provide more or less collateral than accurately needed due to decimal precision not being considered. This can lead to inefficiencies, where users either over-collateralize (locking up more funds than necessary) or under-collateralize (posing a risk to the system's financial stability). Modify the CalcMinCollateral function to account for collateral_decimals. This involves adjusting the final collateral amount by the decimal precision of the collateral asset, ensuring the calculation reflects the actual minimum required collateral in terms of the asset's smallest unit.

Remediation Plan

SOLVED: The **Elys Network team** solved this issue by adjusting decimals.

Remediation Hash

https://github.com/elys-network/elys/pull/375/commits/c148ea189cee549e3c9aa61f1c5f7a7defbcbbcd

7.3 ABSENCE OF CREATION FEE IN CREATEPOOL METHOD LEADING TO SPAMMING ATTACKS

// HIGH

Description

Upon review of the **CreatePool** method within the Keeper module, a notable gap has been identified: the method does not implement a creation fee for new pools. This absence can potentially lead to spamming attacks, as malicious actors may exploit it to create numerous pools with minimal cost, cluttering the system and possibly degrading its performance. The lack of a creation fee for pools could lead to an influx of low-quality or spam pools being created on the platform.

```
func (k Keeper) CreatePool(ctx sdk.Context, msg *types.MsgCreatePool) (uint64, error) {
    sender := msg.GetSigners()[0]

    entry, found := k.assetProfileKeeper.GetEntry(ctx, ptypes.BaseCurrency)
    if !found {
        return 0, errorsmod.Wrapf(assetprofiletypes.ErrAssetProfileNotFound, "asset %s not found", ptypes.BaseCurrency)
    }
    baseCurrency := entry.Denom

    // If the fee denom is empty, set it to the base currency
    if msg.PoolParams.FeeDenom == "" {
        msg.PoolParams.FeeDenom = baseCurrency
    }

    // Get the next pool ID and increment the pool ID counter
    // Create the pool with the given pool ID
    poolId := k.GetNextPoolId(ctx)
    pool, err := types.NewBalancerPool(poolId, *msg.PoolParams, msg.PoolAssets, ctx.BlockTime())
    if err != nil {
        return 0, err
    }

    if err := pool.Validate(poolId); err != nil {
        return 0, err
    }

    address, err := sdk.AccAddressFromBech32(pool.GetAddress())
    if err != nil {
        return 0, fmt.Errorf("invalid pool address %s", pool.GetAddress())
    }

    // create and save the pool's module account to the account keeper
    if err := utils.CreateModuleAccount(ctx, k.accountKeeper, address); err != nil {
        return 0, fmt.Errorf("creating pool module account for id %d: %w", poolId, err)
    }

    // Run the initialization logic.
    if err := k.InitializePool(ctx, &pool, sender); err != nil {
        return 0, err
    }

    // Send initial liquidity to the pool's address.
    initialPoolLiquidity := msg.InitialLiquidity()
    err = k.bankKeeper.SendCoins(ctx, sender, address, initialPoolLiquidity)
    if err != nil {
        return 0, err
    }

    // Increase liquidty amount
    for _, asset := range msg.PoolAssets {
        k.RecordTotalLiquidityIncrease(ctx, sdk.Coins{asset.Token})
    }

    // emitCreatePoolEvents(ctx, poolId, msg)
    return pool.GetPoolId(), nil
}
```

Proof of Concept

```
func (suite *KeeperTestSuite) TestMsgServerCreatePool() {
    for _, tc := range []struct {
```

```
        desc              string
        senderInitBalance sdk.Coins
        poolParams        types.PoolParams
        poolAssets        []types.PoolAsset
        expSenderBalance  sdk.Coins
        expTotalLiquidity sdk.Coins
        expLpCommitment   sdk.Coin
        expPass           bool
    }{
        {
            desc:              "zero tvl pool creation",
            senderInitBalance: sdk.Coins{sdk.NewInt64Coin(ptypes.Eden, 1000000), sdk.NewInt64Coin(ptypes.Elys, 1000000)},
            poolParams: types.PoolParams{
                SwapFee:                   sdk.ZeroDec(),
                ExitFee:                   sdk.ZeroDec(),
                UseOracle:                 false,
                WeightBreakingFeeMultiplier: sdk.ZeroDec(),
                WeightBreakingFeeExponent:   sdk.NewDecWithPrec(25, 1), // 2.5
                ExternalLiquidityRatio:      sdk.NewDec(1),
                WeightRecoveryFeePortion:    sdk.NewDecWithPrec(10, 2), // 10%
                ThresholdWeightDifference:   sdk.ZeroDec(),
                FeeDenom:                  ptypes.BaseCurrency,
            },
            poolAssets: []types.PoolAsset{
                {
                    Token:  sdk.NewInt64Coin(ptypes.Eden, 1000000),
                    Weight: sdk.OneInt(),
                },
                {
                    Token:  sdk.NewInt64Coin(ptypes.Elys, 1000000),
                    Weight: sdk.OneInt(),
                },
            },
            expSenderBalance: sdk.Coins{},
            expLpCommitment:  sdk.NewCoin("amm/pool/1", sdk.NewInt(100).Mul(types.OneShare)),
            expPass:          true,
        },
        {
            desc:              "positive tvl pool creation",
            senderInitBalance: sdk.Coins{sdk.NewInt64Coin(ptypes.Eden, 1000000), sdk.NewInt64Coin(ptypes.BaseCurrency,
1000000)},
            poolParams: types.PoolParams{
                SwapFee:                   sdk.ZeroDec(),
                ExitFee:                   sdk.ZeroDec(),
                UseOracle:                 false,
                WeightBreakingFeeMultiplier: sdk.ZeroDec(),
                WeightBreakingFeeExponent:   sdk.NewDecWithPrec(25, 1), // 2.5
                ExternalLiquidityRatio:      sdk.NewDec(1),
                WeightRecoveryFeePortion:    sdk.NewDecWithPrec(10, 2), // 10%
                ThresholdWeightDifference:   sdk.ZeroDec(),
                FeeDenom:                  ptypes.BaseCurrency,
            },
            poolAssets: []types.PoolAsset{
                {
                    Token:  sdk.NewInt64Coin(ptypes.Eden, 1000000),
                    Weight: sdk.OneInt(),
                },
                {
                    Token:  sdk.NewInt64Coin(ptypes.BaseCurrency, 1000000),
                    Weight: sdk.OneInt(),
                },
            },
            expSenderBalance: sdk.Coins{},
            expLpCommitment:  sdk.NewCoin("amm/pool/1", sdk.NewInt(2).Mul(types.OneShare)),
            expPass:          true,
        },
        {
            desc:              "not enough balance to create pool",
            senderInitBalance: sdk.Coins{sdk.NewInt64Coin(ptypes.Eden, 1000000)},
            poolParams: types.PoolParams{
                SwapFee:                   sdk.ZeroDec(),
                ExitFee:                   sdk.ZeroDec(),
                UseOracle:                 false,
                WeightBreakingFeeMultiplier: sdk.ZeroDec(),
                WeightBreakingFeeExponent:   sdk.NewDecWithPrec(25, 1), // 2.5
                ExternalLiquidityRatio:      sdk.NewDec(1),
                WeightRecoveryFeePortion:    sdk.NewDecWithPrec(10, 2), // 10%
                ThresholdWeightDifference:   sdk.ZeroDec(),
                FeeDenom:                  ptypes.BaseCurrency,
            },
            poolAssets: []types.PoolAsset{
                {
                    Token:  sdk.NewInt64Coin(ptypes.Eden, 1000000),
                    Weight: sdk.OneInt(),
                },
                {
                    Token:  sdk.NewInt64Coin(ptypes.BaseCurrency, 1000000),
```

```
                    Weight: sdk.OneInt(),
                },
            },
            expSenderBalance: sdk.Coins{},
            expLpCommitment:  sdk.Coin{},
            expPass:          false,
        },
    } {
        suite.Run(tc.desc, func() {
            suite.SetupTest()
            suite.SetupStableCoinPrices()

            // bootstrap accounts
            sender := sdk.AccAddress(ed25519.GenPrivKey().PubKey().Address())

            // bootstrap balances
            err := suite.app.BankKeeper.MintCoins(suite.ctx, minttypes.ModuleName, tc.senderInitBalance)
            suite.Require().NoError(err)
            err = suite.app.BankKeeper.SendCoinsFromModuleToAccount(suite.ctx, minttypes.ModuleName, sender,
tc.senderInitBalance)
            suite.Require().NoError(err)

            // execute function
            msgServer := keeper.NewMsgServerImpl(suite.app.AmmKeeper)
            resp, err := msgServer.CreatePool(
                sdk.WrapSDKContext(suite.ctx),
                &types.MsgCreatePool{
                    Sender:     sender.String(),
                    PoolParams: &tc.poolParams,
                    PoolAssets: tc.poolAssets,
                })
            if !tc.expPass {
                suite.Require().Error(err)
            } else {
                suite.Require().NoError(err)
                suite.Require().Equal(resp.PoolID, uint64(1))

                pools := suite.app.AmmKeeper.GetAllPool(suite.ctx)
                suite.Require().Len(pools, 1)
                suite.Require().Equal(pools[0].PoolId, uint64(1))
                suite.Require().Equal(pools[0].PoolParams, tc.poolParams)
                suite.Require().Equal(pools[0].TotalShares.Amount.String(), tc.expLpCommitment.Amount.String())

                totalWeight := sdk.ZeroInt()
                for _, poolAsset := range tc.poolAssets {
                    totalWeight = totalWeight.Add(poolAsset.Weight)
                }
                suite.Require().Equal(pools[0].TotalWeight.String(),
totalWeight.MulRaw(types.GuaranteedWeightPrecision).String())

                // check balance change on sender
                balances := suite.app.BankKeeper.GetAllBalances(suite.ctx, sender)
                suite.Require().Equal(balances.String(), tc.expSenderBalance.String())

                // check lp token commitment
                commitments := suite.app.CommitmentKeeper.GetCommitments(suite.ctx, sender.String())
                suite.Require().Len(commitments.CommittedTokens, 1)
                suite.Require().Equal(commitments.CommittedTokens[0].Denom, tc.expLpCommitment.Denom)
                suite.Require().Equal(commitments.CommittedTokens[0].Amount.String(), tc.expLpCommitment.Amount.String())
            }
        })
    }
}
```

BVSS

AO:A/AC:M/AX:M/C:N/I:C/A:C/D:N/Y:N/R:N/S:C (7.0)

Recommendation

Introduce a fee for creating new pools. This fee should be set at a level that discourages spamming while remaining reasonable for legitimate users.

Remediation Plan

SOLVED: The **Elys Network team** solved this issue by adding a pool creation fee.

Remediation Hash

https://github.com/elys-network/elys/pull/447/commits/d5d59a802a7d19f94a33cf91bbf77f420528ab25

7.4 INADEQUATE HANDLING OF LARGE WEBASSEMBLY (WASM) FILES ON THE SYSTEM LEADS TO DOS

// HIGH

Description

Inadequate handling of large WebAssembly (WASM) files on the chain can lead to performance issues, slow startup times, and potentially cause system failure. It can cause to DOS.

Proof of Concept

```
echo "Running store code"
RES=$(client tx wasm store artifacts/larger_wasm_file.wasm --from alice -y --output json -b block $TXFLAG)
CODE_ID=$(echo $RES | jq -r '.logs[0].events[-1].attributes[0].value')
echo "Got CODE_ID = $CODE_ID"

echo "Deploying contract"
# swallow output
OUT=$(client tx wasm instantiate $CODE_ID "$INIT" --from alice --label "my first contract" --gas-prices 10$FEE_DENOM --gas
auto --gas-adjustment 1.3 -b block -y --no-admin $NODE --chain-id $CHAIN_ID)
ADDR=$(client query wasm list-contract-by-code $CODE_ID --output json $NODE | jq -r '.contracts[0]')
echo "Could not deploy larger wasm file"
```

BVSS

AO:A/AC:L/AX:L/C:N/I:H/A:H/D:H/Y:N/R:P/S:C (7.0)

Recommendation

To prevent issues related to inadequate handling of large WASM files, increase or override the default max file size limit to allow the system to handle larger WASM files. (**https://github.com/osmosis-labs/osmosis/blob/574233334f7b69fe1a58d2f79ef189dd68d32eee/app/app.go#L183**)

Remediation Plan

SOLVED: The **Elys Network team** solved the issue.

Remediation Hash

https://github.com/elys-network/elys/pull/362/commits/76ac10f8a95f689d62abcf8cad452f8b457f5599

7.5 ABSENCE OF IBC CHANNEL VERIFICATION IN UPDATEENTRY FUNCTION

// MEDIUM

Description

In the provided **UpdateEntry** function, part of a system managing entries in IBC. The function updates an entry with various fields, including **IbcChannelId** and **IbcCounterpartyChannelId**. However, it lacks verification for the correctness and validity of these Inter-Blockchain Communication (IBC) channel identifiers.

```
func (k msgServer) UpdateEntry(goCtx context.Context, msg *types.MsgUpdateEntry) (*types.MsgUpdateEntryResponse, error) {
    if k.authority != msg.Authority {
        return nil, errors.Wrapf(govtypes.ErrInvalidSigner, "invalid authority; expected %s, got %s", k.authority,
msg.Authority)
    }

    ctx := sdk.UnwrapSDKContext(goCtx)

    // Check if the value exists
    entry, isFound := k.GetEntry(ctx, msg.BaseDenom)
    if !isFound {
        return nil, errorsmod.Wrap(sdkerrors.ErrKeyNotFound, "entry not set")
    }

    // Checks if the the msg authority is the same as the current owner
    if msg.Authority != entry.Authority {
        return nil, errorsmod.Wrap(sdkerrors.ErrUnauthorized, "incorrect owner")
    }

    entry = types.Entry{
        Authority:               msg.Authority,
        BaseDenom:               msg.BaseDenom,
        Decimals:                msg.Decimals,
        Denom:                   msg.Denom,
        Path:                    msg.Path,
        IbcChannelId:            msg.IbcChannelId,
        IbcCounterpartyChannelId: msg.IbcCounterpartyChannelId,
        DisplayName:             msg.DisplayName,
        DisplaySymbol:           msg.DisplaySymbol,
        Network:                 msg.Network,
        Address:                 msg.Address,
        ExternalSymbol:          msg.ExternalSymbol,
        TransferLimit:           msg.TransferLimit,
        Permissions:             msg.Permissions,
        UnitDenom:               msg.UnitDenom,
        IbcCounterpartyDenom:    msg.IbcCounterpartyDenom,
        IbcCounterpartyChainId:  msg.IbcCounterpartyChainId,
        CommitEnabled:           msg.CommitEnabled,
        WithdrawEnabled:         msg.WithdrawEnabled,
    }

    k.SetEntry(ctx, entry)

    return &types.MsgUpdateEntryResponse{}, nil
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:C (6.3)

Recommendation

Consider validating IBC channel.

Remediation Plan

SOLVED: The **Elys Network team** solved the issue.

Remediation Hash

https://github.com/elys-network/elys/pull/337/commits/537ccb4b7244760b8e774ea509e8d09b124b8514

7.6 LACK OF BROKER ADDRESS VALIDATION IN MSGUPDATEBROKERADDRESS'S VALIDATEBASIC FUNCTION

// MEDIUM

Description

The **MsgUpdateBrokerAddress** function in the given code is designed to update the broker address in app chain. This function creates an instance of MsgUpdateBrokerAddress with a specified creator and broker address. However, a crucial issue is identified in the associated ValidateBasic method: it does not validate the **BrokerAddress**. Currently, the **ValidateBasic** method only validates the Creator field, overlooking the validation of BrokerAddress. For instance, some swap operations can fail.

BVSS

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:N/S:C (6.3)

Recommendation

Enhance the **ValidateBasic** function to include validation for the BrokerAddress. This validation should ensure that the BrokerAddress is correctly formatted and adheres to the expected address standards.

Remediation Plan

SOLVED: The **Elys Network team** solved the issue.

Remediation Hash

https://github.com/elys-network/elys/pull/337/commits/537ccb4b7244760b8e774ea509e8d09b124b8514

7.7 IMPLEMENT TWAP FOR MORE ACCURATE PRICE RETRIEVAL IN KEEPER METHODS

// MEDIUM

Description

The current implementation of Elys Network Keeper methods (**GetLatestPriceFromAssetAndSource** and **GetLatestPriceFromAnySource**) retrieves the latest price of an asset, either from a specific source or any source. While this provides the most recent price data, it may not accurately reflect the market price due to short-term price fluctuations or potential manipulation. To enhance the reliability of the pricing mechanism, we propose the implementation of Time-Weighted Average Price (TWAP).

```
func (k msgServer) FeedPrice(goCtx context.Context, msg *types.MsgFeedPrice) (*types.MsgFeedPriceResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    feeder, found := k.Keeper.GetPriceFeeder(ctx, msg.Provider)
    if !found {
        return nil, types.ErrNotAPriceFeeder
    }

    if !feeder.IsActive {
        return nil, types.ErrPriceFeederNotActive
    }

    price := types.Price{
        Provider:  msg.Provider,
        Asset:     msg.Asset,
        Price:     msg.Price,
        Source:    msg.Source,
        Timestamp: uint64(ctx.BlockTime().Unix()),
    }

    k.SetPrice(ctx, price)
    return &types.MsgFeedPriceResponse{}, nil
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:H/A:N/D:M/Y:N/R:P/S:C (5.5)

Recommendation

Modify the existing GetPrice function to calculate the TWAP. This might involve fetching multiple price points within a defined time window and computing their average.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.8 JOINPOOLNOSWAP EXHIBITS DISCREPANCY BETWEEN EXPECTED AND ACTUAL CALCULATED SHARE OUTPUTS

// LOW

Description

In the current implementation of JoinPoolNoSwap, the implementation does not consider with diff in the calculations of the expected shares out and calculated shares out. The diff in calculations is at most 1 denom-coin - rounding error. There will be dust.

```
func (k Keeper) JoinPoolNoSwap(
    ctx sdk.Context,
    sender sdk.AccAddress,
    poolId uint64,
    shareOutAmount sdk.Int,
    tokenInMaxs sdk.Coins,
) (tokenIn sdk.Coins, sharesOut sdk.Int, err error) {
    // defer to catch panics, in case something internal overflows.
    defer func() {
        if r := recover(); r != nil {
            tokenIn = sdk.Coins{}
            sharesOut = sdk.Int{}
            err = fmt.Errorf("function JoinPoolNoSwap failed due to internal reason: %v", r)
        }
    }()
    // all pools handled within this method are pointer references, `JoinPool` directly updates the pools
    pool, poolExists := k.GetPool(ctx, poolId)
    if !poolExists {
        return nil, sdk.ZeroInt(), types.ErrInvalidPoolId
    }

    if !pool.PoolParams.UseOracle {
        tokensIn := tokenInMaxs
        if len(tokensIn) != 1 {
            // we do an abstract calculation on the lp liquidity coins needed to have
            // the designated amount of given shares of the pool without performing swap
            neededLpLiquidity, err := pool.GetMaximalNoSwapLPAmount(shareOutAmount)
            if err != nil {
                return nil, sdk.ZeroInt(), err
            }

            // check that needed lp liquidity does not exceed the given `tokenInMaxs` parameter. Return error if so.
            // if tokenInMaxs == 0, don't do this check.
            if tokenInMaxs.Len() != 0 {
                if !(neededLpLiquidity.DenomsSubsetOf(tokenInMaxs)) {
                    return nil, sdk.ZeroInt(), errorsmod.Wrapf(types.ErrLimitMaxAmount, "TokenInMaxs does not include all
the tokens that are part of the target pool,"+
                        " upperbound: %v, needed %v", tokenInMaxs, neededLpLiquidity)
                } else if !(tokenInMaxs.DenomsSubsetOf(neededLpLiquidity)) {
                    return nil, sdk.ZeroInt(), errorsmod.Wrapf(types.ErrDenomNotFoundInPool, "TokenInMaxs includes tokens
that are not part of the target pool,"+
                        " input tokens: %v, pool tokens %v", tokenInMaxs, neededLpLiquidity)
                }
                if !(tokenInMaxs.IsAllGTE(neededLpLiquidity)) {
                    return nil, sdk.ZeroInt(), errorsmod.Wrapf(types.ErrLimitMaxAmount, "TokenInMaxs is less than the needed
LP liquidity to this JoinPoolNoSwap,"+
                        " upperbound: %v, needed %v", tokenInMaxs, neededLpLiquidity)
                }
            }

            tokensIn = neededLpLiquidity
        }

        sharesOut, err = pool.JoinPool(ctx, k.oracleKeeper, k.accountedPoolKeeper, tokensIn)
        if err != nil {
            return nil, sdk.ZeroInt(), err
        }

        // sanity check, don't return error as not worth halting the LP. We know its not too much.
        if sharesOut.LT(shareOutAmount) {
            ctx.Logger().Error(fmt.Sprintf("Expected to JoinPoolNoSwap >= %s shares, actually did %s shares",
                shareOutAmount, sharesOut))
        }

        err = k.applyJoinPoolStateChange(ctx, pool, sender, sharesOut, tokensIn)
```

```
        // Increase liquidty amount
        k.RecordTotalLiquidityIncrease(ctx, tokensIn)

        return tokensIn, sharesOut, err
    }

    // on oracle pool, full tokenInMaxs are used regardless shareOutAmount
    sharesOut, err = pool.JoinPool(ctx, k.oracleKeeper, k.accountedPoolKeeper, tokenInMaxs)
    if err != nil {
        return nil, sdk.ZeroInt(), err
    }

    // sanity check, don't return error as not worth halting the LP. We know its not too much.
    if sharesOut.LT(shareOutAmount) {
        ctx.Logger().Error(fmt.Sprintf("Expected to JoinPoolNoSwap >= %s shares, actually did %s shares",
            shareOutAmount, sharesOut))
    }

    err = k.applyJoinPoolStateChange(ctx, pool, sender, sharesOut, tokenInMaxs)

    // Increase liquidty amount
    k.RecordTotalLiquidityIncrease(ctx, tokenInMaxs)

    return tokenInMaxs, sharesOut, err
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

Consider adding a difference check with the bounds.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.9 BULK COIN SENDS

// LOW

Description

For blockchains that support custom token transfer logic, such as blacklists for specific tokens like USDC, it is crucial to ensure that all token transfers in the Begin/EndBlock stages properly handle and catch potential panics. While this is generally a straightforward task, it is commonly overlooked in one specific context: bulk coin sends.

The Cosmos SDK allows for multiple coins to be transferred in a single function call through its SendCoins function. This function acts as a black box, which means it does not provide the ability to validate each individual token transfer. As a result, developers often overlook the necessity to handle panics within this function.

If a single panic is triggered during a call to SendCoins in the Begin/EndBlock stages, it can lead to a complete halt of the blockchain. This highlights the importance of properly handling panics in bulk coin send operations, as failing to do so can have severe consequences on the chain's availability and functionality.

Therefore, when implementing custom token transfer logic or working with bulk coin sends, it is essential to thoroughly review the code and ensure that all potential panic scenarios are properly caught and handled. This proactive approach can help prevent unexpected chain halts and maintain the stability and reliability of the blockchain.

Example Code Location : https://github.com/elys-network/elys/blob/6a6ec2162613ba8bf7d683162698516e7c6eb7ba/x/perpetual/keeper/handle_funding_fee_distribution.go#L80-L81

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

While one can catch the panic on the entire SendCoins call, this would mean that an attacker can DoS all transfers in the batch. Thus, the solution for these situations is to transfer coins one by one with SendCoin and verify each transfer so that problematic ones can be skipped.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.10 LACK OF SPEC ON THE MODULES

// LOW

Description

The spec file intends to outline the common structure for specifications within this directory. The `assetprofile` - `parameter` - `commitment` - `stablestake` - `oracle` - `accountedpool` - `amm` modules are 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.

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

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

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.11 MISSING ASSET WHITELISTING/DENOM CHECK IN FEEDPRICE METHOD

// LOW

Description

The current implementation of the **FeedPrice** method in the **msgServer** struct lacks an essential security feature: asset whitelisting for the price feeder. This function is designed to update the price of an asset provided by a price feeder, but it does not verify whether the asset being updated is recognized or authorized by the system.

```
func (k msgServer) FeedPrice(goCtx context.Context, msg *types.MsgFeedPrice) (*types.MsgFeedPriceResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    feeder, found := k.Keeper.GetPriceFeeder(ctx, msg.Provider)
    if !found {
        return nil, types.ErrNotAPriceFeeder
    }

    if !feeder.IsActive {
        return nil, types.ErrPriceFeederNotActive
    }

    price := types.Price{
        Provider:  msg.Provider,
        Asset:     msg.Asset,
        Price:     msg.Price,
        Source:    msg.Source,
        Timestamp: uint64(ctx.BlockTime().Unix()),
    }

    k.SetPrice(ctx, price)
    return &types.MsgFeedPriceResponse{}, nil
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

Introduce a mechanism to maintain a whitelist of approved assets. The FeedPrice method should include a check to ensure that the asset provided in msg.Asset is part of this whitelist.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.12 INSUFFICIENT VALIDATION IN MSGCREATEENTRY, MSGUPDATEENTRY, AND MSGDELETEENTRY

// LOW

Description

The functions **NewMsgCreateEntry**, **NewMsgUpdateEntry**, and **NewMsgDeleteEntry** are part of a system that manages entries. However, an issue arises in their ValidateBasic methods: there is a lack of comprehensive validation for all the fields in these messages. Currently, the validation only checks the Authority field's format, neglecting other crucial fields like BaseDenom, Denom, Path, IbcChannelId, and others. This insufficient validation can lead to potential operational failures or vulnerabilities, as malformed or invalid data might be processed by the system.

```
func NewMsgUpdateEntry(
    authority string,
    baseDenom string,
    decimals uint64,
    denom string,
    path string,
    ibcChannelId string,
    ibcCounterpartyChannelId string,
    displayName string,
    displaySymbol string,
    network string,
    address string,
    externalSymbol string,
    transferLimit string,
    permissions []string,
    unitDenom string,
    ibcCounterpartyDenom string,
    ibcCounterpartyChainId string,
) *MsgUpdateEntry {
    return &MsgUpdateEntry{
        Authority:                authority,
        BaseDenom:                baseDenom,
        Decimals:                 decimals,
        Denom:                    denom,
        Path:                     path,
        IbcChannelId:             ibcChannelId,
        IbcCounterpartyChannelId: ibcCounterpartyChannelId,
        DisplayName:              displayName,
        DisplaySymbol:            displaySymbol,
        Network:                  network,
        Address:                  address,
        ExternalSymbol:           externalSymbol,
        TransferLimit:            transferLimit,
        Permissions:              permissions,
        UnitDenom:                unitDenom,
        IbcCounterpartyDenom:     ibcCounterpartyDenom,
        IbcCounterpartyChainId:   ibcCounterpartyChainId,
    }
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:C (3.1)

Recommendation

Extend the ValidateBasic method in each message type to include validation checks for all relevant fields. These checks should ensure that the fields are correctly formatted and meet the expected standards.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.13 VULNERABILITIES IN COSMOS SDK V0.47.4 AFFECTING ELYS NETWORK

// LOW

Description

Elys Network, which utilizes the Cosmos SDK version 0.47.4, is exposed to multiple vulnerabilities identified within the specified version of the SDK. The issues encompass a range of security flaws that could potentially impact the stability, security, and integrity of the Elys Network. The vulnerabilities are documented in the following GitHub advisories:

- **GHSA-4j93-fm92-rp4m**: Describes a critical flaw that could allow for unintended behavior or access.
- **GHSA-2557-x9mg-76w8**: Details vulnerabilities that may lead to denial of service (DoS) attacks or affect the network's operational integrity.
- **GHSA-95rx-m9m5-m94v**: Outlines issues that could potentially be exploited to compromise the confidentiality, integrity, or availability of the network.
- **GHSA-86h5-xcpx-cfqc**: Highlights flaws that might allow malicious actors to impact the network adversely.

BVSS

AO:A/AC:L/AX:L/C:N/I:M/A:N/D:N/Y:N/R:P/S:C (3.1)

Recommendation

Promptly upgrade to the latest version of the Cosmos SDK that addresses these vulnerabilities. Ensure thorough testing of the upgraded version in a test environment before deploying it to the mainnet to confirm compatibility and stability.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

7.14 UTILIZE BLOCK HEIGHT IN PRICE FEEDING

// LOW

Description

In the current implementation of the `FeedPrice` function, the timestamp for the price feed is derived from the block time by converting it to Unix time. However, relying solely on the block time may not be sufficient in certain scenarios, as it can potentially lead to issues such as:

1. **Time Manipulation Attacks**: If an attacker can manipulate the block time, they may be able to feed incorrect prices by adjusting the timestamp.
2. **Timestamp Collisions**: If multiple price feeds are submitted within the same Unix timestamp, there could be potential collisions or overwriting of prices, leading to data loss or inconsistencies.
3. **Lack of Determinism**: Using only the block time introduces a non-deterministic element, as the block time can vary slightly between nodes due to network latency or other factors.

To address these potential issues and improve the security and reliability of the price feeding mechanism, it is recommended to utilize both the block time and the block height in the price feed message.

```
func (k msgServer) FeedPrice(goCtx context.Context, msg *types.MsgFeedPrice) (*types.MsgFeedPriceResponse, error) {
    ctx := sdk.UnwrapSDKContext(goCtx)

    feeder, found := k.Keeper.GetPriceFeeder(ctx, msg.Provider)
    if !found {
        return nil, types.ErrNotAPriceFeeder
    }

    if !feeder.IsActive {
        return nil, types.ErrPriceFeederNotActive
    }

    price := types.Price{
        Provider:  msg.Provider,
        Asset:     msg.Asset,
        Price:     msg.Price,
        Source:    msg.Source,
        Timestamp: uint64(ctx.BlockTime().Unix()),
    }

    k.SetPrice(ctx, price)
    return &types.MsgFeedPriceResponse{}, nil
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:L/D:N/Y:N/R:P/S:C (3.1)

Recommendation

Modify the `MsgFeedPrice` message to include the block height in addition to the timestamp.

Remediation Plan

SOLVED: The **Elys Network team** solved the issue by implementing the suggested recommendation.

Remediation Hash

https://github.com/elys-network/elys/pull/394/commits/c782c0bd2141a5e805a14ffa25a25adf97be8dbc

7.15 POTENTIAL DENIAL OF SERVICE (DOS) VULNERABILITY IN POWAPPROX FUNCTION

// LOW

Description

The Elys project uses a function called `PowApprox` to calculate approximate values for the exponentiation operation (`a^b`). This function is implemented using a Taylor series approximation, and it uses a while loop to iterate until the terms being added are sufficiently small. However, the implementation does not have a bound on the maximum number of iterations, which can potentially lead to a Denial of Service (DoS) vulnerability.

The issue arises when specific values of `a` and `b` are chosen, such as `a = 1.999999999999999` and `b = 0.1`. In these cases, the `PowApprox` function can take an excessive number of iterations (over two million iterations on an M1 processor) to converge, resulting in a prolonged computation time of over 800 milliseconds.

This long runtime is not accounted for in the gas costs of transactions that use the `PowApprox` function. As a result, an attacker can craft a transaction that calls `PowApprox` with carefully chosen values, causing the Elys nodes to spend an inordinate amount of time computing this function without having to pay a proportional gas cost.

```
// Contract: 0 < base <= 2
// 0 <= exp < 1.
func PowApprox(base sdk.Dec, exp sdk.Dec, precision sdk.Dec) sdk.Dec {
    if !base.IsPositive() {
        panic(fmt.Errorf("base must be greater than 0"))
    }

    if exp.IsZero() {
        return sdk.OneDec()
    }

    // Common case optimization
    // Optimize for it being equal to one-half
    if exp.Equal(one_half) {
        output, err := base.ApproxSqrt()
        if err != nil {
            panic(err)
        }
        return output
    }
    // TODO: Make an approx-equal function, and then check if exp * 3 = 1, and do a check accordingly

    // We compute this via taking the maclaurin series of (1 + x)^a
    // where x = base - 1.
    // The maclaurin series of (1 + x)^a = sum_{k=0}^{infty} binom(a, k) x^k
    // Binom(a, k) takes the natural continuation on the first parameter, namely that
    // Binom(a, k) = N/D, where D = k!, and N = a(a-1)(a-2)...(a-k+1)
    // Next we show that the absolute value of each term is less than the last term.
    // Note that the change in term n's value vs term n + 1 is a multiplicative factor of
    // v_n = x(a - n) / (n+1)
    // So if |v_n| < 1, we know that each term has a lesser impact on the result than the last.
    // For our bounds on |x| < 1, |a| < 1,
    // it suffices to see for what n is |v_n| < 1,
    // in the worst parameterization of x = 1, a = -1.
    // v_n = |(-1 + epsilon - n) / (n+1)|
    // So |v_n| is always less than 1, as n ranges over the integers.
    //
    // Note that term_n of the expansion is 1 * prod_{i=0}^{n-1} v_i
    // The error if we stop the expansion at term_n is:
    // error_n = sum_{k=n+1}^{infty} term_k
    // At this point we further restrict a >= 0, so 0 <= a < 1.
    // Now we take the _INCORRECT_ assumption that if term_n < p, then
    // error_n < p.
    // This assumption is obviously wrong.
    // However our usages of this function don't use the full domain.
    // With a > 0, |x| << 1, and p sufficiently low, perhaps this actually is true.

    // TODO: Check with our parameterization
    // TODO: If theres a bug, balancer is also wrong here :thonk:

    base = base.Clone()
    x, xneg := AbsDifferenceWithSign(base, one)
    term := sdk.OneDec()
    sum := sdk.OneDec()
```

```
    negative := false

    a := exp.Clone()
    bigK := sdk.NewDec(0)
    // TODO: Document this computation via taylor expansion
    for i := int64(1); term.GTE(precision); i++ {
        // At each iteration, we need two values, i and i-1.
        // To avoid expensive big.Int allocation, we reuse bigK variable.
        // On this line, bigK == i-1.
        c, cneg := AbsDifferenceWithSign(a, bigK)
        // On this line, bigK == i.
        bigK.Set(sdk.NewDec(i)) // TODO: O(n) bigint allocation happens
        term.MulMut(c).MulMut(x).QuoMut(bigK)

        // a is mutated on absDifferenceWithSign, reset
        a.Set(exp)

        if term.IsZero() {
            break
        }
        if xneg {
            negative = !negative
        }

        if cneg {
            negative = !negative
        }

        if negative {
            sum.SubMut(term)
        } else {
            sum.AddMut(term)
        }
    }
    return sum
}
```

Proof of Concept

```
// calculate a^b
// assumption: a is between 0 and 2, b is between 0 and 1
fn PowApprox(a,b) {
  total <- 1
  i <- 0
  term <- 1
  const precision = 0.00000001
  // (the real implementation took precision as a function parameter rather than a constant)
  while abs(term) >= precision {
    i <- i + 1
    term <- term * ((b-(i-1)) / i) * (a-1)
    total <- total + term
  }
  return total
}
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:L/D:N/Y:N/R:P/S:C (2.0)

Recommendation

Implement a maximum iteration limit or a timeout mechanism in the `PowApprox` function to prevent excessive computation times.

Remediation Plan

RISK ACCEPTED: The **Elys Network team** accepted the risk of the issue.

References

osmosis-labs/osmosis/pull/6627

7.16 INCREASE PRECISION OF INITIAL POOL SHARES TO MATCH INDUSTRY STANDARDS

// INFORMATIONAL

Description

We've identified a potential issue with the precision of the initial shares allocated to the first liquidity provider (LP) of a pool. Currently, the protocol initializes a pool with 10^18 shares. However, this might not offer enough precision, especially in scenarios where the initial liquidity provider is a significant entity (like the project team) and contributes a substantial amount of liquidity. For context, Balancer, a well-known protocol in the DeFi space, initializes its pools with a higher precision, specifically 100 * 10^18 shares (reference: Balancer's BConst.sol). This higher level of precision allows for more granular control and accuracy in the distribution of pool shares, which is particularly important in the DeFi ecosystem where large and fractional values are common.

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

Increase the precision of initial pool shares from 10^18 to 100 * 10^18.

Remediation Plan

SOLVED: The **Elys Network team** solved the issue by adjusting the share amount.

Remediation Hash

https://github.com/elys-network/elys/pull/447/commits/743916095819033ee6dcb476398fbf2df279a6e8

7.17 MISSING USAGE DESCRIPTION FOR ALL TRANSACTION COMMANDS CLI

// INFORMATIONAL

Description

All the transaction and query commands for all the modules are missing a long message to provide description about their usage, which will be helpful for users and external developers.

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

It is recommended to specify a long message for all transaction commands. Each command should provide a description of how to use the command correctly.

Remediation Plan

ACKNOWLEDGED: The **Elys Network team** acknowledged the issue.

7.18 IMPLEMENT FEE MARKET INTEGRATED INTO CONSENSUS LAYER

// INFORMATIONAL

Description

During periods of high network activity, maintaining liveness and usability becomes challenging without a robust fee market mechanism in place. While there are various approaches, the Ethereum Improvement Proposal (EIP-1559) has proven to be an effective default solution. However, implementing it directly in the mempool, as some blockchains have done, is suboptimal compared to integrating the fee market directly into the consensus layer.
The superior long-term solution is to leverage the `ABCI 2.0` protocol and integrate the fee market into the consensus layer. This approach offers several advantages, including improved reliability, scalability, and adaptability to changing network conditions. By leveraging `ABCI 2.0`, blockchains can seamlessly adopt the fee market mechanism without the need for extensive modifications or hard forks.

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

Conduct thorough testing and simulations to ensure the fee market mechanism operates as intended under various network conditions, including periods of high usage and congestion.

Remediation Plan

ACKNOWLEDGED: The **Elys Network team** acknowledged the issue.

7.19 LACK OF IBC RATE-LIMITTING IMPLEMENTATION

// INFORMATIONAL

Description

Rate limits are safety controls that disable certain functionality of a system when a pre-defined threshold is surpassed. In this context, we categorize rate limit mechanisms that don't require external trigger invocation, as the tripping conditions are defined in the protocol. As such, rate limit thresholds tend to be simpler and numeric. (Osmosis Implementation : **https://commonwealth.im/osmosis/discussion/8543-set-ibc-rate-limits**)

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

Implement rate-limiting on the IBC like an Osmosis.

Remediation Plan

ACKNOWLEDGED: The **Elys Network team** acknowledged the issue.

7.20 ARBITRARY TOKEN TRANSFER LEADS TO CHAIN HALT

// INFORMATIONAL

Description

In blockchains that allow for arbitrary token transfer hooks and execute them during the Begin/EndBlock stages, a potential vulnerability exists where an attacker can create a token that triggers an infinitely looping CosmWasm contract. When this malicious token is transferred during the next block's BeginBlock, it can cause the entire chain to halt due to the unintended infinite loop.

This issue highlights the importance of properly handling and limiting the execution of arbitrary code during critical phases of the blockchain's operation, such as the BeginBlocker and EndBlocker operations. Failing to impose appropriate constraints can lead to Denial of Service (DoS) attacks and severe disruptions to the chain's availability and functionality.

The root cause of this vulnerability lies in the lack of proper gas metering or execution time limits for the arbitrary code executed during token transfers. An attacker can exploit this by crafting a token that triggers an infinite loop or resource-intensive computation, effectively locking up the node's resources and preventing further block processing.

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

Implement gas metering or execution time limits for all arbitrary code executed during token transfers or other critical blockchain operations. This can be achieved by leveraging the existing gas metering mechanisms in the Cosmos SDK or implementing custom time-based limits.

Remediation Plan

ACKNOWLEDGED: The **Elys Network team** acknowledged the issue.

7.21 KEEP SLIPPAGE EXPIRATION DATE AS A GOVERNANCE PARAMETER

// INFORMATIONAL

Description

In the provided code snippet, the `ClearOutdatedSlippageTrack` function is responsible for deleting expired slippage tracks. However, the expiration date is currently hardcoded as a constant value of 7 days (86400 * 7 seconds). This hardcoded value limits the flexibility of the system and may not be suitable for all scenarios or deployment environments.
Having a fixed expiration date for slippage tracks can lead to potential issues, such as:

1. **Lack of Adaptability**: Different projects or use cases may require different retention periods for slippage tracks, depending on their specific requirements or analysis needs.
2. **Limited Governance**: The expiration date is hardcoded in the system, preventing the community or governance mechanisms from adjusting or fine-tuning this parameter based on changing circumstances or feedback.
3. **Maintenance Challenges**: If the desired expiration period needs to be changed in the future, it would require modifying the codebase and potentially deploying a new version of the software, which can be a time-consuming and error-prone process.

Code Location : https://github.com/elys-network/elys/blob/6a6ec2162613ba8bf7d683162698516e7c6eb7ba/x/amm/keeper/abci.go#L174-L175

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

Create a new governance parameter that represents the slippage expiration date (e.g., `SlippageTrackExpirationDays`). This parameter should be stored in the module's parameter store and initialized with a sensible default value (e.g., 7 days).

Remediation Plan

SOLVED: The **Elys Network team** solved the issue by adding suggested recommendations.

Remediation Hash

https://github.com/elys-network/elys/pull/447/commits/743916095819033ee6dcb476398fbf2df279a6e8

7.22 SILENT ERROR IN RECORDWITHDRAWVALIDATORCOMMISSION FUNCTION

// INFORMATIONAL

Description

In the `RecordWithdrawValidatorCommission` function within the keeper, there is a potential issue where the function does not return an error in case of failure.

https://github.com/elys-network/elys/blob/7a20f784857ecda997c6cc55dbd52068f13926c1/x/incentive/keeper/keeper_withdraw.go#L191-L192

```
    // Eden
    unclaimed := commitments.GetRewardUnclaimedForDenom(ptypes.Eden)
    if !unclaimed.IsZero() {
        err = k.cmk.RecordWithdrawValidatorCommission(ctx, delegator, validator, ptypes.Eden, unclaimed)
    }

    // EdenB
    unclaimed = commitments.GetRewardUnclaimedForDenom(ptypes.EdenB)
    if !unclaimed.IsZero() {
        err = k.cmk.RecordWithdrawValidatorCommission(ctx, delegator, validator, ptypes.EdenB, unclaimed)
    }
```

BVSS

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:P/S:C (0.0)

Recommendation

Modify the `RecordWithdrawValidatorCommission` function to handle errors gracefully and return them consistently.

Remediation Plan

ACKNOWLEDGED: The **Elys Network team** acknowledged the issue.

8. AUTOMATED TESTING

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were **staticcheck**, **gosec**, **semgrep**, **unconvert**, **codeql** and **nancy**. After Halborn verified all the code 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.

Gosec

```
[/elys-0.29.2/x/tokenomics/module.go:75] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    74: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 75:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    76: }



[/elys-0.29.2/x/stablestake/module.go:73] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    72: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 73:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    74: }



[/elys-0.29.2/x/perpetual/module.go:75] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    74: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 75:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    76: }



[/elys-0.29.2/x/perpetual/migrations/v5_migration.go:18] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    17:     for _, mtp := range m.keeper.GetAllMTPs(ctx) {
  > 18:         m.keeper.DestroyMTP(ctx, mtp.Address, mtp.Id)
    19:     }



[/elys-0.29.2/x/perpetual/migrations/v5_migration.go:14] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    13:     params := types.NewParams()
  > 14:     m.keeper.SetParams(ctx, &params)
    15:



[/elys-0.29.2/x/perpetual/migrations/v5_migration.go:10] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    9:  // reset store
  > 10:     m.keeper.ResetStore(ctx)
    11:



[/elys-0.29.2/x/perpetual/migrations/v4_migration.go:18] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    17:     for _, mtp := range m.keeper.GetAllMTPs(ctx) {
  > 18:         m.keeper.DestroyMTP(ctx, mtp.Address, mtp.Id)
    19:     }



[/elys-0.29.2/x/perpetual/migrations/v4_migration.go:14] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    13:     params := types.NewParams()
  > 14:     m.keeper.SetParams(ctx, &params)
    15:



[/elys-0.29.2/x/perpetual/migrations/v4_migration.go:10] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    9:  // reset store
  > 10:     m.keeper.ResetStore(ctx)
    11:



[/elys-0.29.2/x/perpetual/migrations/v3_migration.go:10] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    9:  params := types.NewParams()
  > 10:     m.keeper.SetParams(ctx, &params)
    11:     return nil
```

```
[/elys-0.29.2/x/perpetual/migrations/v2_migration.go:10] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    9:  params := types.NewParams()
  > 10:     m.keeper.SetParams(ctx, &params)
    11:     return nil



[/elys-0.29.2/x/perpetual/keeper/open_short_process.go:83] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    82:     // Set MTP
  > 83:     k.OpenShortChecker.SetMTP(ctx, mtp)
    84:



[/elys-0.29.2/x/perpetual/keeper/open_short_process.go:77] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    76:     // Update consolidated collateral amount
  > 77:     k.OpenShortChecker.CalcMTPConsolidateCollateral(ctx, mtp, baseCurrency)
    78:



[/elys-0.29.2/x/perpetual/keeper/open_long_process.go:103] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    102:    // Set MTP
  > 103:    k.OpenLongChecker.SetMTP(ctx, mtp)
    104:



[/elys-0.29.2/x/perpetual/keeper/open_long_process.go:97] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    96:     // Update consolidated collateral amount
  > 97:     k.OpenLongChecker.CalcMTPConsolidateCollateral(ctx, mtp, baseCurrency)
    98:



[/elys-0.29.2/x/perpetual/genesis.go:35] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    34:     // this line is used by starport scaffolding # genesis/module/init
  > 35:     k.SetParams(ctx, &genState.Params)
    36: }



[/elys-0.29.2/x/parameter/module.go:75] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    74: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 75:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    76: }



[/elys-0.29.2/x/oracle/module.go:76] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    75: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 76:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    77: }



[/elys-0.29.2/x/leveragelp/module.go:75] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    74: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 75:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    76: }



[/elys-0.29.2/x/leveragelp/migrations/v2_migration.go:10] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    9:  params := types.NewParams()
  > 10:     m.keeper.SetParams(ctx, &params)
    11:     return nil



[/elys-0.29.2/x/leveragelp/genesis.go:32] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    31:     // this line is used by starport scaffolding # genesis/module/init
  > 32:     k.SetParams(ctx, &genState.Params)
    33: }



[/elys-0.29.2/x/incentive/module.go:75] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
```

```
    74: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 75:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    76: }



[/elys-0.29.2/x/incentive/keeper/hooks_staking.go:16] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    15:     // Create an entity in commitment module
  > 16:     k.cmk.BeforeDelegationCreated(ctx, delAddr.String(), valAddr.String())
    17:



[/elys-0.29.2/x/incentive/keeper/hooks_commitment.go:14] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    13: func (k Keeper) EdenUncommitted(ctx sdk.Context, creator string, amount sdk.Coin) {
  > 14:     k.BurnEdenBFromEdenUncommitted(ctx, creator, amount.Amount)
    15: }



[/elys-0.29.2/x/commitment/module.go:74] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    73: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 74:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    75: }



[/elys-0.29.2/x/burner/module.go:74] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    73: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 74:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    75: }



[/elys-0.29.2/x/assetprofile/module.go:74] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    73: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 74:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    75: }



[/elys-0.29.2/x/amm/module.go:75] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    74: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 75:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    76: }



[/elys-0.29.2/x/amm/keeper/update_pool_for_swap.go:98] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    97:     if k.hooks != nil {
  > 98:         k.hooks.AfterSwap(ctx, sender, pool, tokensIn, tokensOut)
    99:     }



[/elys-0.29.2/x/amm/keeper/msg_server_update_pool_params.go:33] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH,
Severity: LOW)
    32:     pool.PoolParams = poolParams
  > 33:     k.SetPool(ctx, pool)
    34:



[/elys-0.29.2/x/amm/keeper/msg_server_feed_multiple_external_liquidity.go:92] - G104 (CWE-703): Errors unhandled.
(Confidence: HIGH, Severity: LOW)
    91:         pool.PoolParams.ExternalLiquidityRatio = elRatio
  > 92:         k.SetPool(ctx, pool)
    93:     }



[/elys-0.29.2/x/amm/keeper/denom_liquidity.go:89] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    88:
  > 89:         k.IncreaseDenomLiquidity(ctx, coin.Denom, coin.Amount)
    90:     }



[/elys-0.29.2/x/amm/genesis.go:13] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    12:     for _, elem := range genState.PoolList {
  > 13:         k.SetPool(ctx, elem)
    14:     }



[/elys-0.29.2/x/accountedpool/module.go:74] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
```

```
    73: func (AppModuleBasic) RegisterGRPCGatewayRoutes(clientCtx client.Context, mux *runtime.ServeMux) {
  > 74:     types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
    75: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:38] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    37: func (k Keeper) AfterAmmSwap(ctx sdk.Context, ammPool ammtypes.Pool, perpetualPool perpetualtypes.Pool) {
  > 38:     k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
    39: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:33] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    32: func (k Keeper) AfterAmmExitPool(ctx sdk.Context, ammPool ammtypes.Pool, perpetualPool perpetualtypes.Pool) {
  > 33:     k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
    34: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:28] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    27: func (k Keeper) AfterAmmJoinPool(ctx sdk.Context, ammPool ammtypes.Pool, perpetualPool perpetualtypes.Pool) {
  > 28:     k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
    29: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:23] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    22: func (k Keeper) AfterAmmPoolCreated(ctx sdk.Context, ammPool ammtypes.Pool) {
  > 23:     k.InitiateAccountedPool(ctx, ammPool)
    24: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:18] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    17: func (k Keeper) AfterPerpetualPositionClosed(ctx sdk.Context, ammPool ammtypes.Pool, perpetualPool
perpetualtypes.Pool) {
  > 18:     k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
    19: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:14] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    13: func (k Keeper) AfterPerpetualPositionModified(ctx sdk.Context, ammPool ammtypes.Pool, perpetualPool
perpetualtypes.Pool) {
  > 14:     k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
    15: }



[/elys-0.29.2/x/accountedpool/keeper/hooks_perpetual.go:10] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
    9: func (k Keeper) AfterPerpetualPositionOpen(ctx sdk.Context, ammPool ammtypes.Pool, perpetualPool perpetualtypes.Pool)
{
  > 10:     k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
    11: }
```

Staticcheck

```
accountedpool/keeper/accounted_pool_initiate.go:25:2: should replace loop with accountedPool.PoolAssets =
append(accountedPool.PoolAssets, ammPool.PoolAssets...) (S1011)
accountedpool/keeper/accounted_pool_update_test.go:47:2: should replace loop with accountedPool.PoolAssets =
append(accountedPool.PoolAssets, ammPool.PoolAssets...) (S1011)
accountedpool/keeper/keeper.go:42:83: sdk.Int is deprecated: Functionality of this package has been moved to it's own
module: cosmossdk.io/math (SA1019)
accountedpool/module_simulation.go:45:63: simtypes.WeightedProposalContent is deprecated: Use WeightedProposalMsg instead.
(SA1019)
accountedpool/types/accounted_pool.pb.go:34:14: github_com_cosmos_cosmos_sdk_types.Int is deprecated: Functionality of this
package has been moved to it's own module: cosmossdk.io/math (SA1019)
accountedpool/types/errors.go:11:33: sdkerrors.Register is deprecated: functionality of this package has been moved to it's
own module: (SA1019)
accountedpool/types/errors.go:12:33: sdkerrors.Register is deprecated: functionality of this package has been moved to it's
own module: (SA1019)
accountedpool/types/pool_asset.pb.go:30:9: github_com_cosmos_cosmos_sdk_types.Int is deprecated: Functionality of this
package has been moved to it's own module: cosmossdk.io/math (SA1019)
accountedpool/types/query.pb.gw.go:16:2: "github.com/golang/protobuf/descriptor" is deprecated: See the
```

"google.golang.org/protobuf/reflect/protoreflect" package for how to obtain an EnumDescriptor or MessageDescriptor in order to programatically interact with the protobuf type system. (SA1019)
accountedpool/types/query.pb.gw.go:17:2: "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.org/protobuf/proto" package instead. (SA1019)
accountedpool/types/query.pb.gw.go:32:9: descriptor.ForMessage is deprecated: Not all concrete message types satisfy the Message interface. Use MessageDescriptorProto instead. If possible, the calling code should be rewritten to use protobuf reflection instead. See package "google.golang.org/protobuf/reflect/protoreflect" for details. (SA1019)
amm/client/wasm/query_balance_of_denom.go:12:20: func (*Querier).queryBalanceOfDenom is unused (U1000)
amm/keeper/apply_exit_pool_state_change.go:10:109: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/apply_exit_pool_state_change_test.go:8:2: package "github.com/elys-network/elys/x/amm/types" is being imported more than once (ST1019)
    amm/keeper/apply_exit_pool_state_change_test.go:9:2: other import of "github.com/elys-network/elys/x/amm/types"
amm/keeper/apply_join_pool_state_change.go:8:109: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/calc_in_route_spot_price.go:10:5: should omit nil check; len() for []*github.com/elys-network/elys/x/amm/types.SwapAmountInRoute is defined as zero (S1009)
amm/keeper/calc_out_route_spot_price.go:10:5: should omit nil check; len() for []*github.com/elys-network/elys/x/amm/types.SwapAmountOutRoute is defined as zero (S1009)
amm/keeper/create_elys_multihop_expected_swap_outs.go:14:6: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/create_elys_multihop_expected_swap_outs.go:15:24: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/elys_routed_multihop.go:41:2: should use 'return poolIds[0] != poolIds[1]' instead of 'if poolIds[0] == poolIds[1] { return false }; return true' (S1008)
amm/keeper/keeper_exit_pool.go:13:16: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/keeper_join_pool_no_swap.go:22:17: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/keeper_join_pool_no_swap.go:24:33: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/keeper_join_pool_no_swap.go:29:16: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/keeper_swap_exact_amount_in_test.go:20:23: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/keeper_swap_exact_amount_out_test.go:19:23: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/msg_server_exit_pool_test.go:19:21: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/msg_server_join_pool_test.go:18:21: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/msg_server_swap_by_denom_test.go:19:21: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/msg_server_swap_exact_amount_in_test.go:20:21: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/msg_server_swap_exact_amount_out_test.go:18:21: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/msg_server_test.go:13:6: func setupMsgServer is unused (U1000)
amm/keeper/pool.go:146:2: redundant return statement (S1023)
amm/keeper/pool_share.go:38:2: this value of entry is never used (SA4006)
amm/keeper/pool_share.go:92:104: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/pool_share_test.go:11:2: package "github.com/elys-network/elys/x/amm/types" is being imported more than once (ST1019)
    amm/keeper/pool_share_test.go:12:2: other import of "github.com/elys-network/elys/x/amm/types"
amm/keeper/route_exact_amount_in_test.go:20:23: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/route_exact_amount_out_test.go:20:23: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/keeper/update_pool_for_swap.go:21:4: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/module_simulation.go:70:63: simtypes.WeightedProposalContent is deprecated: Use WeightedProposalMsg instead. (SA1019)
amm/types/calc_exit_pool.go:11:139: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/calc_exit_pool.go:18:2: should merge variable declaration with assignment on next line (S1021)
amm/types/calc_exit_pool.go:82:129: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/calc_exit_pool.go:90:2: should merge variable declaration with assignment on next line (S1021)
amm/types/calc_in_amt_given_out.go:25:3: this value of err is never used (SA4006)
amm/types/calc_out_amt_given_in.go:44:3: this value of err is never used (SA4006)
amm/types/denom_liquidity.pb.go:30:12: github_com_cosmos_cosmos_sdk_types.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/expected_keepers.go:55:51: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/hooks.go:10:104: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/hooks.go:13:81: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/hooks.go:35:126: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/hooks.go:41:103: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/keys.go:102:30: variable in loop condition never changes (SA4008)
amm/types/message_exit_pool.go:13:90: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/message_join_pool.go:13:90: sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math (SA1019)

```
amm/types/pool.go:142:35: sdk.Int is deprecated: Functionality of this package has been moved to it's own module:
cosmossdk.io/math (SA1019)
amm/types/pool.go:146:44: sdk.Int is deprecated: Functionality of this package has been moved to it's own module:
cosmossdk.io/math (SA1019)
amm/types/pool.go:235:58: sdk.Int is deprecated: Functionality of this package has been moved to it's own module:
cosmossdk.io/math (SA1019)
amm/types/pool.pb.go:35:20: github_com_cosmos_cosmos_sdk_types.Int is deprecated: Functionality of this package has been
moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/pool_asset.pb.go:30:9: github_com_cosmos_cosmos_sdk_types.Int is deprecated: Functionality of this package has
been moved to it's own module: cosmossdk.io/math (SA1019)
amm/types/pool_calc_join_pool_no_swap_shares.go:21:68: sdk.Int is deprecated: Functionality of this package has been moved
to it's own module: cosmossdk.io/math (SA1019)
amm/types/pool_calc_join_pool_no_swap_shares.go:82:72: sdk.Int is deprecated: Functionality of this package has been moved
to it's own module: cosmossdk.io/math (SA1019)
amm/types/pool_calc_join_pool_shares.go:10:116: sdk.Int is deprecated: Functionality of this package has been moved to it's
own module: cosmossdk.io/math (SA1019)
amm/types/pool_calc_join_pool_shares.go:10:136: sdk.Int is deprecated: Functionality of this package has been moved to it's
own module: cosmossdk.io/math (SA1019)
amm/types/pool_calc_join_pool_shares.go:36:77: sdk.Int is deprecated: Functionality of this package has been moved to it's
own module: cosmossdk.io/math (SA1019)
amm/types/pool_exit_pool.go:7:124: sdk.Int is deprecated: Functionality of this package has been moved to it's own module:
cosmossdk.io/math (SA1019)
amm/types/pool_exit_pool.go:22:87: sdk.Int is deprecated: Functionality of this package has been moved to it's own module:
cosmossdk.io/math (SA1019)
```

Errcheck

```
accountedpool/keeper/accounted_pool.go:40:22:   defer iterator.Close()
accountedpool/keeper/accounted_pool_update_test.go:93:25:   apk.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/keeper/hooks_perpetual.go:10:23:  k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/keeper/hooks_perpetual.go:14:23:  k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/keeper/hooks_perpetual.go:18:23:  k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/keeper/hooks_perpetual.go:23:25:  k.InitiateAccountedPool(ctx, ammPool)
accountedpool/keeper/hooks_perpetual.go:28:23:  k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/keeper/hooks_perpetual.go:33:23:  k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/keeper/hooks_perpetual.go:38:23:  k.UpdateAccountedPool(ctx, ammPool, perpetualPool)
accountedpool/module.go:74:34:  types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
amm/genesis.go:13:12:   k.SetPool(ctx, elem)
amm/keeper/abci_test.go:288:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/abci_test.go:289:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/batch_processing.go:41:22:   defer iterator.Close()
amm/keeper/batch_processing.go:56:22:   defer iterator.Close()
amm/keeper/batch_processing.go:84:22:   defer iterator.Close()
amm/keeper/batch_processing.go:99:22:   defer iterator.Close()
amm/keeper/calc_in_route_spot_price_test.go:91:29:  suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/calc_in_route_spot_price_test.go:92:29:  suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/calc_out_route_spot_price_test.go:91:29: suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/calc_out_route_spot_price_test.go:92:29: suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/calc_swap_estimation_by_denom_test.go:91:29: suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/calc_swap_estimation_by_denom_test.go:92:29: suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/denom_liquidity.go:40:22:    defer iterator.Close()
amm/keeper/denom_liquidity.go:89:27:    k.IncreaseDenomLiquidity(ctx, coin.Denom, coin.Amount)
amm/keeper/keeper_test.go:122:12:   k.SetPool(ctx, pool)
amm/keeper/msg_server_feed_multiple_external_liquidity.go:92:12:    k.SetPool(ctx, pool)
amm/keeper/msg_server_join_pool_test.go:174:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/msg_server_swap_by_denom_test.go:144:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/msg_server_swap_by_denom_test.go:145:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/msg_server_swap_exact_amount_in_test.go:177:31:  suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/msg_server_swap_exact_amount_in_test.go:178:31:  suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/msg_server_swap_exact_amount_in_test.go:271:29:  suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/msg_server_swap_exact_amount_out_test.go:174:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/msg_server_swap_exact_amount_out_test.go:175:31: suite.app.AmmKeeper.SetPool(suite.ctx, pool2)
amm/keeper/msg_server_update_pool_params.go:33:11:  k.SetPool(ctx, pool)
amm/keeper/oracle_pool_slippage_track.go:40:22: defer iterator.Close()
amm/keeper/oracle_pool_slippage_track.go:56:22: defer iterator.Close()
amm/keeper/oracle_pool_slippage_track.go:75:22: defer iterator.Close()
amm/keeper/pool.go:41:22:   defer iterator.Close()
amm/keeper/pool.go:56:22:   defer iterator.Close()
amm/keeper/pool.go:135:22:  defer iterator.Close()
amm/keeper/pool_test.go:35:17:  keeper.SetPool(ctx, items[i])
amm/keeper/pool_test.go:96:17:  keeper.SetPool(ctx, item)
amm/keeper/route_exact_amount_in_test.go:176:31:    suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/route_exact_amount_out_test.go:174:31:   suite.app.AmmKeeper.SetPool(suite.ctx, pool)
amm/keeper/update_pool_for_swap.go:98:20:   k.hooks.AfterSwap(ctx, sender, pool, tokensIn, tokensOut)
amm/module.go:75:34:    types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
assetprofile/keeper/entry.go:33:22: defer iterator.Close()
assetprofile/keeper/entry.go:57:22: defer iterator.Close()
assetprofile/module.go:74:34:   types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
burner/keeper/history.go:57:22: defer iterator.Close()
burner/module.go:74:34: types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
commitment/keeper/commitments.go:25:22: defer iterator.Close()
commitment/keeper/commitments.go:72:22: defer iterator.Close()
commitment/keeper/deposit_liquid_tokens_test.go:53:35:  keeper.DepositLiquidTokensClaimed(ctx, ptypes.Eden, sdk.NewInt(100),
```

```
creator.String())
commitment/module.go:74:34: types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
epochs/keeper/epoch_infos.go:41:22: defer iterator.Close()
incentive/keeper/elys_staked.go:40:22:  defer iterator.Close()
incentive/keeper/hooks_commitment.go:14:32: k.BurnEdenBFromEdenUncommitted(ctx, creator, amount.Amount)
incentive/keeper/hooks_staking.go:16:31:    k.cmk.BeforeDelegationCreated(ctx, delAddr.String(), valAddr.String())
incentive/keeper/keeper_apr_per_pool_test.go:92:29: ik.UpdateLPRewardsUnclaimed(ctx, lpIncentive)
incentive/keeper/keeper_apr_per_pool_test.go:114:29:    ik.UpdateLPRewardsUnclaimed(ctx, lpIncentive)
incentive/keeper/keeper_withdraw_test.go:126:46:    app.CommitmentKeeper.BeforeDelegationCreated(ctx, delegator,
valAddress.String())
incentive/module.go:75:34:  types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
leveragelp/genesis.go:32:13:    k.SetParams(ctx, &genState.Params)
leveragelp/keeper/begin_blocker_test.go:30:13:  k.SetParams(suite.ctx, &params)
leveragelp/keeper/begin_blocker_test.go:56:13:  k.SetParams(suite.ctx, &params)
leveragelp/keeper/params_test.go:15:13: k.SetParams(ctx, &params)
leveragelp/keeper/pool.go:20:22:    defer iterator.Close()
leveragelp/keeper/position_close_test.go:36:29: suite.app.AmmKeeper.SetPool(suite.ctx, ammtypes.Pool{
leveragelp/keeper/position_open_test.go:31:29:  suite.app.AmmKeeper.SetPool(suite.ctx, ammtypes.Pool{
leveragelp/keeper/query_params_test.go:16:18:   keeper.SetParams(ctx, &params)
leveragelp/keeper/utils_test.go:19:13:  k.SetParams(suite.ctx, &params)
leveragelp/keeper/utils_test.go:29:13:  k.SetParams(suite.ctx, &params)
leveragelp/keeper/utils_test.go:95:13:  k.SetParams(suite.ctx, &params)
leveragelp/keeper/utils_test.go:123:29: suite.app.AmmKeeper.SetPool(suite.ctx, ammtypes.Pool{
leveragelp/migrations/v2_migration.go:10:20:    m.keeper.SetParams(ctx, &params)
leveragelp/module.go:75:34: types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
oracle/keeper/asset_info.go:39:22:  defer iterator.Close()
oracle/keeper/price.go:32:22:   defer iterator.Close()
oracle/keeper/price.go:46:22:   defer iterator.Close()
oracle/keeper/price.go:68:22:   defer iterator.Close()
oracle/keeper/price_feeder.go:40:22:    defer iterator.Close()
oracle/module.go:76:34: types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
parameter/module.go:75:34:  types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
perpetual/genesis.go:35:13: k.SetParams(ctx, &genState.Params)
perpetual/keeper/get_amm_pool_test.go:66:23:    app.AmmKeeper.SetPool(ctx, expectedPool)
perpetual/keeper/get_best_pool_test.go:66:23:   app.AmmKeeper.SetPool(ctx, pool)
perpetual/keeper/open_long_process.go:97:48:    k.OpenLongChecker.CalcMTPConsolidateCollateral(ctx, mtp, baseCurrency)
perpetual/keeper/open_long_process.go:103:26:   k.OpenLongChecker.SetMTP(ctx, mtp)
perpetual/keeper/open_short_process.go:77:49:   k.OpenShortChecker.CalcMTPConsolidateCollateral(ctx, mtp, baseCurrency)
perpetual/keeper/open_short_process.go:83:27:   k.OpenShortChecker.SetMTP(ctx, mtp)
perpetual/keeper/params_test.go:15:13:  k.SetParams(ctx, &params)
perpetual/keeper/pool.go:20:22: defer iterator.Close()
perpetual/keeper/query_params_test.go:16:18:    keeper.SetParams(ctx, &params)
perpetual/keeper/reset_store.go:22:19:  defer iter.Close()
perpetual/migrations/v2_migration.go:10:20: m.keeper.SetParams(ctx, &params)
perpetual/migrations/v3_migration.go:10:20: m.keeper.SetParams(ctx, &params)
perpetual/migrations/v4_migration.go:10:21: m.keeper.ResetStore(ctx)
perpetual/migrations/v4_migration.go:14:20: m.keeper.SetParams(ctx, &params)
perpetual/migrations/v4_migration.go:18:22: m.keeper.DestroyMTP(ctx, mtp.Address, mtp.Id)
perpetual/migrations/v5_migration.go:10:21: m.keeper.ResetStore(ctx)
perpetual/migrations/v5_migration.go:14:20: m.keeper.SetParams(ctx, &params)
perpetual/migrations/v5_migration.go:18:22: m.keeper.DestroyMTP(ctx, mtp.Address, mtp.Id)
perpetual/module.go:75:34:  types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
stablestake/keeper/debt.go:50:22:   defer iterator.Close()
stablestake/module.go:73:34:    types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
tokenomics/keeper/airdrop.go:40:22: defer iterator.Close()
tokenomics/keeper/time_based_inflation.go:51:22:    defer iterator.Close()
tokenomics/module.go:75:34: types.RegisterQueryHandlerClient(context.Background(), mux, types.NewQueryClient(clientCtx))
transferhook/keeper/swap_test.go:243:31:    suite.app.AmmKeeper.SetPool(suite.ctx, pool)
```

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.