// HALBORN

# Sifchain - CLP Update

## Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **June 1st, 2022 - June 23rd, 2022**

Visit: **Halborn.com**

# DOCUMENT REVISION HISTORY

| VERSION | MODIFICATION | DATE | AUTHOR |
|---|---|---|---|
| 0.1 | Document Creation | 06/20/2022 | Gokberk Gulgun |
| 0.2 | Document Updates | 06/22/2022 | John Saigle |
| 0.3 | Document Updates | 06/23/2022 | Chris Meistre |
| 0.4 | Draft Review | 06/23/2022 | Gabi Urrutia |
| 1.0 | Remediation Plan | 07/08/2022 | Chris Meistre |
| 1.1 | Remediation Plan Review | 07/08/2022 | Gabi Urrutia |

# CONTACTS

| CONTACT | COMPANY | EMAIL |
|---|---|---|
| Rob Behnke | Halborn | Rob.Behnke@halborn.com |
| Steven Walbroehl | Halborn | Steven.Walbroehl@halborn.com |
| Gabi Urrutia | Halborn | Gabi.Urrutia@halborn.com |
| John Saigle | Halborn | John.Saigle@halborn.com |
| Chris Meistre | Halborn | Chris.Meistre@halborn.com |

# EXECUTIVE OVERVIEW

## 1.1 INTRODUCTION

Sifchain engaged Halborn to conduct a security audit on the CLP Cosmos Module Updates beginning on June 1st, 2022 and ending on June 23rd, 2022 . The security assessment was scoped to the Cosmos CLP module changes provided to the Halborn team.

## 1.2 AUDIT SUMMARY

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

The purpose of this audit to achieve the following:

- Ensure that the refactoring of the mathematical functionality CLP module within Sifchain is sound.
- Identify potential security issues with the recent changes.

**In summary, Halborn identified few security risks that were accepted and addressed by the Sifchain team.**

## 1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the Sifchain. 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 audit consisted of the following stages:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (staticcheck, gosec, errcheck, golangci-lint and semgrep ).
- Manual Assessment for discovering security vulnerabilities on codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis on CLP functions and data types.

## RISK METHODOLOGY:

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

**RISK SCALE - LIKELIHOOD**

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

**RISK SCALE - IMPACT**

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

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

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


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

## 1.4 SCOPE

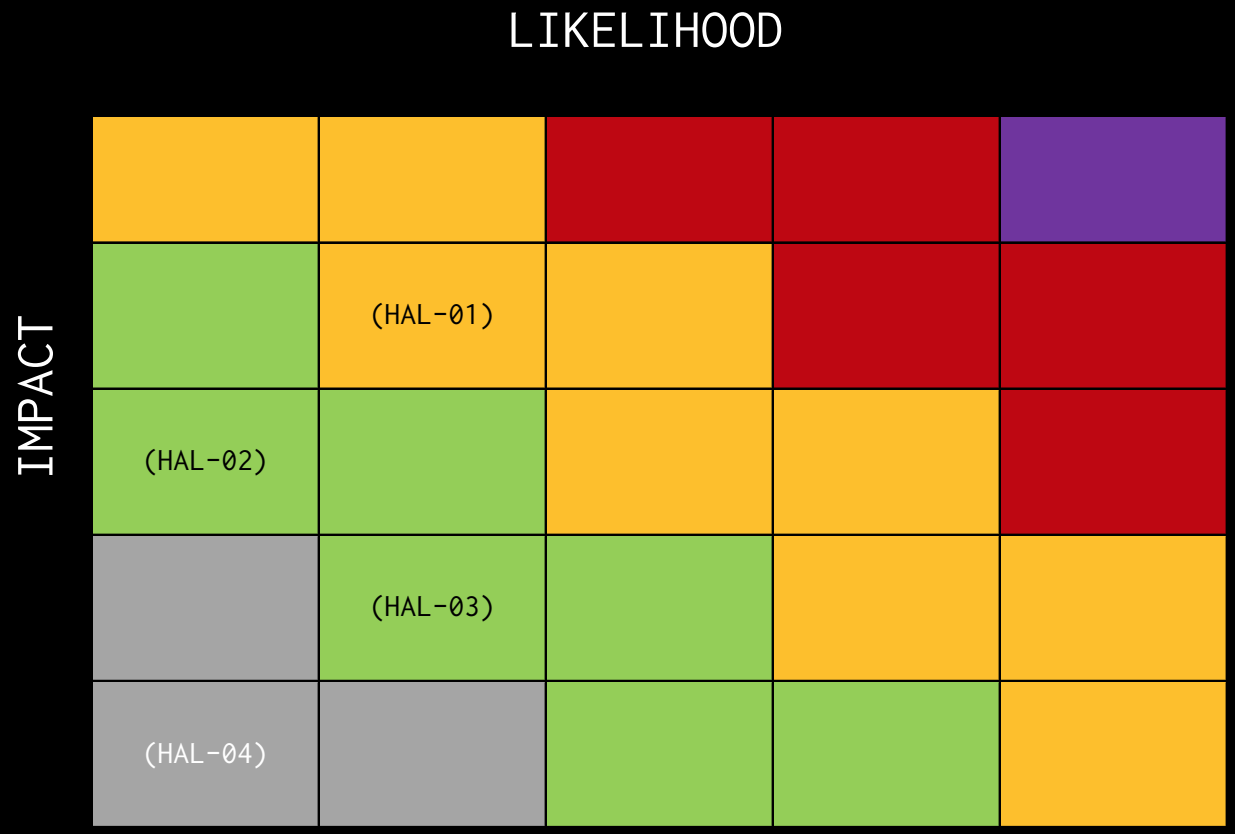
IN-SCOPE:
The security assessment was scoped to Sifchain/sifnode repository.

Branch
Refactor Commit ID
Diff Commit ID

OUT-OF-SCOPE:
External libraries.

# 2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

| CRITICAL | HIGH | MEDIUM | LOW | INFORMATIONAL |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 1 |



# EXECUTIVE OVERVIEW

| SECURITY ANALYSIS | RISK LEVEL | REMEDIATION DATE |
|---|---|---|
| HAL01 - VALIDATION ERROR IN ALLOW-LIST FOR TOKENS THAT CAN BE USED IN LIQUIDITY POOLS | Medium | SOLVED - 07/08/2022 |
| HAL02 - LACK OF ERROR HANDLING | Low | RISK ACCEPTED |
| HAL03 - MISSING FUNCTION PARAMETER VALIDATION | Low | RISK ACCEPTED |
| HAL04 - RE-IMPLEMENTATION OF CORE MATHEMATICAL FUNCTION | Informational | ACKNOWLEDGED |

# FINDINGS & TECH DETAILS

## 3.1 (HAL-01) VALIDATION ERROR IN ALLOW-LIST FOR TOKENS THAT CAN BE USED IN LIQUIDITY POOLS - MEDIUM

Description:

The CLP module allows users to create liquidity pools by pairing an external token with the native token rowan. The external token's symbol is checked against an allow-list of pre-approved tokens. Additionally, it is forbidden to submit rowan as the symbol for the external token to avoid pairing the native token with itself.

However, it is possible to use the symbol Rowan to create a liquidity pool. (Note the capital 'R' instead of the lowercase 'r'.)

This can be done in the case where a user can add a balance of a fake Rowan token to their account that meets the minimum threshold required to create a pool, as well as a sufficient amount of rowan to form the other half of the pair (and pay transaction fees).

If these conditions are met, a user who creates a pool using the unapproved Rowan token can swap this fake asset for valid assets. This could result in negative financial impact for the protocol and users, as swaps using this token could allow a user to siphon value from the pool as well as manipulate prices.

Several features of Sifchain allow for the impact of this issue to be amplified:

1. **Asymmetric pool creation:** this feature means that new pools do not need to be created with an equal value of native and external tokens. Instead, a user could create a pool consisting mostly of the fake token Rowan and a minimal value of rowan. This reduces the upfront amount of rowan an attacker would need to invest when creating a pool and allows them to amplify the volume of their swaps for a lower cost.

2. **Cross-chain support**: an attacker could swap the fake Rowan tokens for ceth which represents a wrapped Ethereum token. If they withdraw ceth into their account, they could then transfer the Ethereum value out of the Sifchain ecosystem and reduce liquidity for other users.

**Examples of correct validation:**

Listing 1

```
# Example 1: Using native token "rowan" as the external asset
$ sifnoded tx clp create-pool \
    --symbol rowan \
    --nativeAmount 100000000000000000 \
    --externalAmount 400000000000000000000000 \
    --from sif --keyring-backend test \
    --fees 100000000000000000rowan \
    --chain-id localnet \
    --broadcast-mode block \
    -y

Error: External Asset cannot be rowan: asset is invalid
[...]

# Example 2: Using an invalid token "faketoken" as the external
↳ asset
$ sifnoded tx clp create-pool \
    --symbol faketoken \
    --nativeAmount 100000000000000000 \
    --externalAmount 400000000000000000000000 \
    --from sif --keyring-backend test \
    --fees 100000000000000000rowan \
    --chain-id localnet \
    --broadcast-mode block \
    -y

[...]
raw_log: 'failed to execute message; message index: 0: Token not
↳ supported by sifchain'
[...]
```

**Creating a Rowan liquidity pool**

**Listing 2**

```
$ sifnoded tx clp create-pool \
    --symbol Rowan \
    --nativeAmount 100000000000000000 \
    --externalAmount 400000000000000000000000 \
    --from sif --keyring-backend test \
    --fees 100000000000000000rowan \
    --chain-id localnet \
    --broadcast-mode block \
    -y
[...]
# The following message indicates that a pool was successfully created
logs:
- events:
  - attributes:
    - key: receiver
      value: sif1pjm228rsgwqf23arkx7lm9ypkyma7mzr3y2n85
    - key: amount
      value: 400000000000000000000000Rowan,100000000000000000rowan
    type: coin_received
# At this stage, the allow-list has been subverted and 'Rowan' can be swapped for any other assets in the Sifchain liquidity pools as though it was an approved token
```

Code Location:

These two code locations contain functionality that compare strings. In combination they allow the fake token Rowan to appear as a valid entry in the allow-list of tokens and at the same time not register as the native token rowan.

x/clp/types/msgs.go

A **case-sensitive** comparison is used which detects the string rowan but not Rowan so no error is triggered.

**Listing 3: (Line 461)**

```
459 func (m MsgCreatePool) ValidateBasic() error {
460     [...]
461     if m.ExternalAsset.Equals(GetSettlementAsset()) {
462         return sdkerrors.Wrap(ErrInValidAsset, "External Asset
 ↳ cannot be rowan")
463     }
```

x/tokenregistry/keeper/keeper.go

A **case-insensitive** comparison is used (strings.EqualFold) which considers Rowan to be equal to rowan. This allows Rowan to appear as a valid entry in the allow-list.

**Listing 4: (Line 106)**

```
103 func (k keeper) GetEntry(wl types.Registry, denom string) (*types.
 ↳ RegistryEntry, error) {
104     for i := range wl.Entries {
105         e := wl.Entries[i]
106         if e != nil && strings.EqualFold(e.Denom, denom) {
107             return wl.Entries[i], nil
108         }
109     }
110     return nil, errors.Wrap(errors.ErrKeyNotFound, "registry entry
 ↳  not found")
111 }
```

Risk Level:

**Likelihood - 2**
**Impact - 4**

Recommendation:

Further validation should be performed on the symbol used for external assets in new liquidity pools. Providing any variation on the word “rowan” should give the same error message as the faketoken example

above. Ensure that case-sensitive string comparisions are use everywhere in the codebase especially with security-sensitive parameters.

Remediation Plan:

**SOLVED**: The code was updated to do a strict string equality check in b72bf38.

## 3.2 (HAL-02) LACK OF ERROR HANDLING - LOW

Description:

Some sections of the codebase contain calls to functions which may throw errors. However, no error checking is in place.

Failure to handle error conditions may result in unexpected behaviour, information disclosure (such as stack traces), and denial-of-service in the case where the lack of error handling causes a node to crash.

Code Location:

**Example 1**: Panic when bit length is too large.

**Listing 5**

```
$ sifnoded tx clp swap --from sif --keyring-backend test --
↳ sentSymbol cdash --receivedSymbol ceth --sentAmount 9x115 --
↳ minReceivingAmount 0 --fees 1000000000000000000rowan --chain-id
↳ localnet -y
panic: bit length 386 greater than 256
```

**Example 2**: Panic when sentAmount is negative.

**Listing 6**

```
$ sifnoded tx clp swap --from sif --keyring-backend test --
↳ sentSymbol cdash --receivedSymbol ceth --sentAmount -1 --
↳ minReceivingAmount 0 --fees 1000
00000000000000rowan --chain-id localnet -y

panic: non-positive integer
```

**Example 3**: Panic when submitting governance proposal.

**Listing 7**

```
sifnoded tx gov submit-proposal param-change ./scripts/proposal.
↳ json \
--from sif --keyring-backend test \
--fees 100000rowan \
--chain-id localnet \
--broadcast-mode block \
-y
[...]
raw_log: 'panic message redacted to hide potentially sensitive
↳ system info: panic'
```

**Example 4**: Linting tool errcheck disabled.

x/clp/keeper/migrations.go, Lines 77-78

**Listing 8**

```
// nolint:errcheck
m.keeper.SetPool(ctx, &pool)
```

**Example 5**: Comment claims no error can be thrown but the function itself can throw errors.

x/clp/keeper/pmtp.go, Lines 88-89

**Listing 9**

```
// ignore error since it will always be nil
_ = k.SetPool(ctx, pool)
```

x/clp/keeper/pool.go, Lines 12-23

**Listing 10**

```
func (k Keeper) SetPool(ctx sdk.Context, pool *types.Pool) error {
    if !pool.Validate() {
        return types.ErrUnableToSetPool
    }
    store := ctx.KVStore(k.storeKey)
```

```
17      key, err := types.GetPoolKey(pool.ExternalAsset.Symbol, types.
↳ GetSettlementAsset().Symbol)
18      if err != nil {
19          return err
20      }
21      store.Set(key, k.cdc.MustMarshal(pool))
22      return nil
23  }
```

Risk Level:

**Likelihood - 1**
**Impact - 3**

Recommendation:

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

Remediation Plan:

**RISK ACCEPTED**: Since the generated panics are from the client side or not possible to produce within the context of their function, the risk has been accepted by the Sifchain Team.

## 3.3 (HAL-03) MISSING FUNCTION PARAMETER VALIDATION - LOW

Description:

We found that there are two functions (calculateSlipAdjustment and calcSwap) that are not checking all the parameters passed to it in order to prevent a division by zero error. This kind of error can lead to a "panic" where the node may disclose sensitive information, behave in unexpected ways, or crash.

The functions do no validation on their arguments. Instead, they expect that these values have already been checked before the function is called.

In all cases, the values to these functions are being properly validated before the functions are called. However, it can lead to a problem in the future if another developer is unaware of this convention and uses the functions incorrectly. The requirement for the parameters to be checked before the function is called is not documented within the code.

In calculateSlipAdjustment if r + R = 0 or a + A = 0 there will be a division by zero error on line 273.

In calcSwap if x + X = 0 there will be a division by zero error on line 333.

Code Location:

x/clp/keeper/pool.go, Lines 256-278

**Listing 11: (Line 273)**

```
256 func calculateSlipAdjustment(R, A, r, a *big.Int) *
 ↳ slipAdjustmentValues {
257     var denominator, rPlusR, aPlusA big.Int
258     rPlusR.Add(r, R)
259     aPlusA.Add(a, A)
260     denominator.Mul(&rPlusR, &aPlusA)
```

```
261
262     var RTimesa, rTimesA, nominator big.Int
263     RTimesa.Mul(R, a)
264     rTimesA.Mul(r, A)
265     nominator.Sub(&RTimesa, &rTimesA)
266
267     var one, nom, denom, slipAdjustment big.Rat
268     one.SetInt64(1)
269
270     nom.SetInt(&nominator)
271     denom.SetInt(&denominator)
272
273     slipAdjustment.Quo(&nom, &denom)
274     slipAdjustment.Abs(&slipAdjustment)
275     slipAdjustment.Sub(&one, &slipAdjustment)
276
277     return &slipAdjustmentValues{slipAdjustment: &slipAdjustment,
 ↳ RTimesa: &RTimesa, rTimesA: &rTimesA}
278 }
```

x/clp/keeper/pool.go, Lines 322-336

**Listing 12: (Line 333)**

```
322 func calcSwap(x, X, Y *big.Int) big.Rat {
323     var s, d, d2, d3 big.Int
324     var numerator, denominator, y big.Rat
325
326     s.Add(X, x)     // s = X + x
327     d.Mul(&s, &s)   // d = (X + x)**2
328     d2.Mul(X, Y)    // d2 = X * Y
329     d3.Mul(x, &d2) // d3 = x * X * Y
330
331     denominator.SetInt(&d)
332     numerator.SetInt(&d3)
333     y.Quo(&numerator, &denominator) // y = d3 / d = (x * X * Y) /
 ↳ (X + x)**2
334
335     return y
336 }
```

Risk Level:

**Likelihood - 2**
**Impact - 2**

Recommendation:

It is recommended that functions contain code to validate all parameters passed to them. Otherwise, any prerequisites should be documented within the code.

Remediation Plan:

**RISK ACCEPTED**: Since both the functions are private and are only called from a single source, the risk has been accepted by the Sifchain Team.

## 3.4 (HAL-04) RE-IMPLEMENTATION OF CORE MATHEMATICAL FUNCTION - INFORMATIONAL

Description:

The file x/clp/keeper/pureCalculation.go contains a function Abs() that calculates the absolute value of a signed integer. While no problems were detected in the logic of this code, it is generally advisable to avoid re-implementing fundamental mathematical functions. Instead, it is best to use standard libraries that have been examined and tested by many open-source developers over the course of years. Doing so can also reduce the complexity of the project, as it removes the need to maintain and test functionality that is already solved.

Both CosmosSDK and Go's "bigInt" library contain implementations of an absolute value function that could be used instead.

Code Location:

x/clp/keeper/pureCalculation.go, Lines 61-66

**Listing 13**

```
1 func Abs(a int16) uint16 {
2     if a < 0 {
3         return uint16(-a)
4     }
5     return uint16(a)
6 }
```

Risk Level:

**Likelihood - 1**
**Impact - 1**

Recommendation:

Use the equivalent function in the CosmosSDK or Go library instead of re-implementing it.

Remediation Plan:

**ACKNOWLEDGED**: Since the available `Abs()` function does not work with the integer type that is in use, the risk has been accepted by the Sifchain Team.

# AUTOMATED TESTING

Description:

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

Semgrep - Security Analysis Output Sample:

**Listing 14: Rule Set**

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

# AUTOMATED TESTING

## Semgrep Results:

```
Findings:

  x/clp/types/querier.pb.gw.go
     go.gorilla.security.audit.handler-assignment-from-multiple-sources.handler-assignment-from-
     multiple-sources
        Variable val, ok is assigned from two different sources: 'pathParams["symbol"]' and
        'pathParams["lp_address"]'. Make sure this is intended, as this could cause logic bugs if
        they are treated as they are the same object.
        Details: https://sg.run/gL3y

       137┆ val, ok = pathParams["symbol"]
       138┆ if !ok {
       139┆     return nil, metadata, status.Errorf(codes.InvalidArgument, "missing parameter %s", "symbol")
       140┆ }
       141┆
       142┆ protoReq.Symbol, err = runtime.String(val)
       143┆
       144┆ if err != nil {
       145┆     return nil, metadata, status.Errorf(codes.InvalidArgument, "type mismatch, parameter: %s, error: %v", "symbol", err)
       146┆ }
       147┆
       148┆ val, ok = pathParams["lp_address"]
          ┆----------------------------------------
       175┆ val, ok = pathParams["symbol"]
       176┆ if !ok {
       177┆     return nil, metadata, status.Errorf(codes.InvalidArgument, "missing parameter %s", "symbol")
       178┆ }
       179┆
       180┆ protoReq.Symbol, err = runtime.String(val)
       181┆
       182┆ if err != nil {
       183┆     return nil, metadata, status.Errorf(codes.InvalidArgument, "type mismatch, parameter: %s, error: %v", "symbol", err)
       184┆ }
       185┆
       186┆ val, ok = pathParams["lp_address"]
```

```
Findings:

  x/clp/keeper/executors.go
     trailofbits.go.invalid-usage-of-modified-variable.invalid-usage-of-modified-variable
        Variable `lp` is likely modified and later used on error. In some cases this could result
        in panics due to a nil dereference
        Details: https://sg.run/WWQ2

       106┆ lp, err := k.GetLiquidityProvider(ctx, msg.ExternalAsset.Symbol, msg.Signer)
       107┆ if err != nil {
       108┆     lp = k.CreateLiquidityProvider(ctx, msg.ExternalAsset, lpUnits, addr)
       109┆     ctx.EventManager().EmitEvents(sdk.Events{
       110┆             sdk.NewEvent(
       111┆                     types.EventTypeCreateLiquidityProvider,
       112┆                     sdk.NewAttribute(types.AttributeKeyLiquidityProvider, lp.String()),
       113┆                     sdk.NewAttribute(types.AttributeKeyHeight, strconv.FormatInt(ctx.BlockHeight(), 10)),
       114┆             ),
       115┆     })
       116┆     lpUnits = sdk.ZeroUint()
       117┆ }


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

        52┆ protoReq.Symbol, err = runtime.String(val)
          ┆----------------------------------------
        79┆ protoReq.Symbol, err = runtime.String(val)
          ┆----------------------------------------
       142┆ protoReq.Symbol, err = runtime.String(val)
          ┆----------------------------------------
       153┆ protoReq.LpAddress, err = runtime.String(val)
          ┆----------------------------------------
       180┆ protoReq.Symbol, err = runtime.String(val)
          ┆----------------------------------------
       191┆ protoReq.LpAddress, err = runtime.String(val)
          ┆----------------------------------------
       222┆ protoReq.LpAddress, err = runtime.String(val)
          ┆----------------------------------------
       256┆ protoReq.LpAddress, err = runtime.String(val)
          ┆----------------------------------------
       294┆ protoReq.LpAddress, err = runtime.String(val)
          ┆----------------------------------------
       328┆ protoReq.LpAddress, err = runtime.String(val)
          ┆----------------------------------------
       402┆ protoReq.Symbol, err = runtime.String(val)
          ┆----------------------------------------
       436┆ protoReq.Symbol, err = runtime.String(val)
```

# AUTOMATED TESTING

```
x/dispensation/client/cli/tx.go
   trailofbits.go.invalid-usage-of-modified-variable.invalid-usage-of-modified-variable
      Variable `dispensationCount` is likely modified and later used on error. In some cases this
      could result  in panics due to a nil dereference
      Details: https://sg.run/WWQ2

      117  dispensationCount, err := strconv.ParseInt(args[2], 10, 64)
      118  if err != nil {
      119     return fmt.Errorf("invalid dispensation count :%d", dispensationCount)
      120  }


x/ibctransfer/types/mocks/expected_keepers_gen.go
   trailofbits.go.unchecked-type-assertion.unchecked-type-assertion
      Unchecked type assertion.
      Details: https://sg.run/054W

       44  ret0, _ := ret[0].(error)
           |----------------------------------------
       58  ret0, _ := ret[0].(error)
           |----------------------------------------
       72  ret0, _ := ret[0].(error)
           |----------------------------------------
      109  ret0, _ := ret[0].(error)
           |----------------------------------------
      123  ret0, _ := ret[0].(error)
           |----------------------------------------
      137  ret0, _ := ret[0].(error)
           |----------------------------------------
      174  ret0, _ := ret[0].(*types0.MsgTransferResponse)
           |----------------------------------------
      175  ret1, _ := ret[1].(error)


x/tokenregistry/types/mock/keeper_gen.go
   trailofbits.go.unchecked-type-assertion.unchecked-type-assertion
      Unchecked type assertion.
      Details: https://sg.run/054W

       43  ret0, _ := ret[0].(bool)
           |----------------------------------------
       57  ret0, _ := ret[0].(*types.GenesisState)
           |----------------------------------------
       71  ret0, _ := ret[0].(*types.RegistryEntry)
           |----------------------------------------
       72  ret1, _ := ret[1].(error)
           |----------------------------------------
       86  ret0, _ := ret[0].(types.Registry)
           |----------------------------------------
      100  ret0, _ := ret[0].([]types1.ValidatorUpdate)
           |----------------------------------------
      114  ret0, _ := ret[0].(bool)
```

## Gosec - Security Analysis Output Sample:

```
[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/test/test_common.go:125] - G404 (CWE-338): Use of weak random number generator (math/rand instead of crypto/rand) (Confidence: MEDIUM, Severity: HIGH)
    124:        for i := 0; i < numberOfLp; i++ {
  > 125:                externalToken := tokens[rand.Intn(len(tokens))]
    126:                asset := types.NewAsset(TrimFirstRune(externalToken))


[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/test/test_common.go:112] - G404 (CWE-338): Use of weak random number generator (math/rand instead of crypto/rand) (Confidence: MEDIUM, Severity: HIGH)
    111:                // initialize global pseudo random generator
  > 112:                externalToken := tokens[rand.Intn(len(tokens))]
    113:                externalAsset := types.NewAsset(TrimFirstRune(externalToken))


[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/test/test_common.go:228] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
    227:        }
  > 228:        input, err := ioutil.ReadFile(file)
    229:        if err != nil {


[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/client/cli/tx.go:432] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
    431:                            }
  > 432:                            input, err = ioutil.ReadFile(file)
    433:                            if err != nil {


[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/client/cli/tx.go:417] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
    416:                        }
  > 417:                        input, err := ioutil.ReadFile(file)
    418:                        if err != nil {


[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/client/cli/tx.go:67] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
     66:                        }
  > 67:                        input, err := ioutil.ReadFile(file)
     68:                        if err != nil {


[/home/chris/Work/Halborn/AUDIT/SIFCHAIN/sifnode/x/clp/keeper/migrations.go:78] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
     77:                // nolint:errcheck
  > 78:                m.keeper.SetPool(ctx, &pool)
     79:        }
```

## Staticcheck - Security Analysis Output Sample:

```
keeper/msg_server.go:373:2: this value of totalLiquidityFee is never used (SA4006)
types/querier.pb.gw.go:16:2: "github.com/golang/protobuf/descriptor" is deprecated: See the "google.golang.org/protobuf/reflect/protoreflect" package for how to obtai
n an EnumDescriptor or MessageDescriptor in order to programatically interact with the protobuf type system.  (SA1019)
types/querier.pb.gw.go:17:2: "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.org/protobuf/proto" package instead.  (SA1019)
types/querier.pb.gw.go:33:9: 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.  (SA101
9)
types/types.go:8:2: should use 'return p.ExternalAsset.Validate()' instead of 'if !p.ExternalAsset.Validate() { return false }; return true' (S1008)
types/types.go:56:2: should use 'return l.Asset.Validate()' instead of 'if !l.Asset.Validate() { return false }; return true' (S1008)
```

## Unconvert - Security Analysis Output Sample:

```
2022/06/23 08:53:58 internal error: package "fmt" without types was imported from "github.com/Sifchain/sifnode/x/clp"
```

## LGTM - Security Analysis Output Sample:

THANK YOU FOR CHOOSING

// HALBORN