Quicksilver - Airdrop

Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 22nd, 2022 - September 1st, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/26/2022	Chris Meistre
0.2	Document Edits	08/29/2022	Gokberk Gulgun
0.3	Document Edits	08/29/2022	Gokberk Gulgun
0.4	Document Draft Review	08/29/2022	Gabi Urrutia
1.0	Remediation Plan	09/02/2022	Gokberk Gulgun
1.1	Remediation Plan Review	09/02/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Chris Meistre	Halborn	Chris.Meistre@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Quicksilver engaged Halborn to conduct a security audit on their Airdrop module, beginning on August 22nd, 2022 and ending on September 1st, 2022. The security assessment was scoped to the code base provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned three full-time security engineers to audit the security of the modules. 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 module functions are intended.
- Report potential security issues to the Quicksilver Team.

In summary, Halborn identified few security risks that were mostly addressed by the Quicksilver 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 Airdrop module. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of structures and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (staticcheck, gosec, unconvert, LGTM, ineffassign and semgrep).
- Manual Assessment for discovering security vulnerabilities on codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis on modules 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.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

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 ingenuity-build/quicksilver repository.

Commit ID

IN-SCOPE Module :

- x/airdrop

REMEDIATION COMMIT PROVIDED: Commit ID

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	1	2	3	4

LIKELIHOOD

IMPACT

🟨	🟨	🟥 (HAL-02)	🟥	🟪 (HAL-01)
🟩	🟨 (HAL-03)	🟨 (HAL-04)	🟥	🟥
🟩 (HAL-07)	🟩	🟨	🟨	🟥
⬜	🟩 (HAL-06)	🟩 (HAL-05)	🟨	🟨
⬜ (HAL-08) (HAL-09) (HAL-10) (HAL-11)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - ENDBLOCKER IMPLEMENTATION CAN LEADS TO CONSENSUS HALT	Critical	SOLVED - 09/02/2022
HAL-02 - NON-DETERMINISTIC TIME COMPARISON	High	SOLVED - 09/02/2022
HAL-03 - MODULE DOES NOT REGISTER SEVERAL TYPES	Medium	SOLVED - 09/02/2022
HAL-04 - IMPROPER BOUND VALIDATION ON THE DURATION	Medium	RISK ACCEPTED
HAL-05 - ONLY CLI IMPLEMENTED	Low	RISK ACCEPTED
HAL-06 - UNHANDLED ERRORS	Low	RISK ACCEPTED
HAL-07 - LACK OF SIMULATION AND FUZZING ON THE AIRDROP MODULE	Low	RISK ACCEPTED
HAL-08 - DUPLICATED ERROR CHECKS	Informational	SOLVED - 09/02/2022
HAL-09 - ERRORS ARE RETURNED AS NIL	Informational	SOLVED - 09/02/2022
HAL-10 - PANIC IS USED FOR ERROR HANDLING	Informational	ACKNOWLEDGED
HAL-11 - OPEN TODOs	Informational	SOLVED - 09/02/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) ENDBLOCKER IMPLEMENTATION CAN LEADS TO CONSENSUS HALT - CRITICAL

Description:

BeginBlocker and **EndBlocker** are optional methods module developers can implement in their module. They will be triggered at the beginning and at the end of each block, respectively, when the **BeginBlock** and **EndBlock** **ABCI** messages are received from the underlying consensus engine. Making use of panics for error handling in the BeginBlock and EndBlock methods may cause the chain to halt if an error does occur. During the code review, It has been observed that If the chain zone does not have enough tokens, that can leads to chain halt.

Code Location:

x/airdrop/keeper/abci.go, Line 15

Listing 1

```
12 func (k Keeper) EndBlocker(ctx sdk.Context) {
13     for _, zd := range k.UnconcludedAirdrops(ctx) {
14         if err := k.EndZoneDrop(ctx, zd.ChainId); err != nil {
15             panic(err)
16         }
17     }
18 }
```

Scenario:

- Define a genesis with the following configurations.
- At the end block, unconcluded airdrops are checked and coins are distributed according to claim records.
- However, If there is no enough coins are hold by the modules, the chain can halt.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

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

Remediation Plan:

SOLVED: The Quicksilver Team solved this issue by removing the **panic** statement from **EndBlocker** in the following commit.

3.2 (HAL-02) NON-DETERMINISTIC TIME COMPARISON - HIGH

Description:

Making use of time.Now() returns the operating system timestamp. Local clock times are subjective and thus non-deterministic. As there might be small discrepancies amongst the timestamp of various nodes, it might lead to the chain not being able to reach consensus. As an example, previously Cosmos SDK had vulnerability named as **jackfruit** which can be resulted with consensus halt on the **x/authz** module.

Code Location:

x/airdrop/types/airdrop.go, Line 95

Listing 2

```
 95     if cr.ActionsCompleted != nil {
 96         // action enum, completed action
 97         i := 0
 98         sum := uint64(0)
 99         for ae, ca := range cr.ActionsCompleted {
100             // check enum bounds
101             kstr := fmt.Sprintf("ActionsCompleted[%d]", i)
102             if int(ae) >= len(Action_name) {
103                 errors[kstr+" Enum"] = fmt.Errorf("%w, got %d",
  ↳ ErrActionOutOfBounds, ae)
104             }
105             // calc sum
106             sum += ca.ClaimAmount
107             // check completed
108             // CompleteTime should be some significant time (not
  ↳ mere
109             // miliseconds) into the past, thus it should not
  ↳ cause determinism
110             // issues related to clock drift.
111             // If however, this turns out to be problematic the
  ↳ check can be
```

```
            // moved from stateless to stateful to ensure a valid
↳ CompleteTime.
            if ca.CompleteTime.After(time.Now()) {
                errors[kstr+" CompleteTime"] = fmt.Errorf("invalid
↳ spacetime continuum, time is in the future")
            }
            // check claim amount
            if ca.ClaimAmount > cr.MaxAllocation {
                errors[kstr+" ClaimAmount"] = fmt.Errorf("exceeds
↳ max allocation")
            }
            i++
        }
```

Risk Level:

Likelihood - 3
Impact - 5

Recommendation:

It is recommended to use **BlockTime** instead of **time.Now()**.

Remediation Plan:

SOLVED: The Quicksilver Team solved this issue by using the **BlockTime** statement instead of **time.Now()** in the following commit.

3.3 (HAL-03) MODULE DOES NOT REGISTER SEVERAL TYPES - MEDIUM

Description:

In the Cosmos 0.40, **RegisterCodec** method is changed to **RegisterLegacyAminoCodec**. The method registers the amino codec for the module, which is used to **marshal** and **unmarshal** structs to/from **[]byte** in order to persist them in the module's **KVStore**. **RegisterInterface** method registers a module's interface types and their concrete implementations as **proto.Message**. The methods are not implemented in the current airdrop module.

Code Location:

x/airdrop/module.go, Line 48

Listing 3

```
// RegisterCodec registers a legacy amino codec
func (AppModuleBasic) RegisterCodec(cdc *codec.LegacyAmino) {}

// RegisterLegacyAminoCodec registers a legacy amino codec
func (AppModuleBasic) RegisterLegacyAminoCodec(cdc *codec.
↳ LegacyAmino) {}

// RegisterInterfaces registers the module's interface types.
func (a AppModuleBasic) RegisterInterfaces(_ cdctypes.
↳ InterfaceRegistry) {}
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

Ensure that all types are registered on the module.

Remediation Plan:

SOLVED: The Quicksilver Team solved this issue by registering all necessity codes in the following commit.

3.4 (HAL-04) IMPROPER BOUND VALIDATION ON THE DURATION - MEDIUM

Description:

The Duration property of a ZoneDrop does not contain any upper/lower bound validation checks. It is only lower bounded by **0** on the implementation.

Code Location:

x/airdrop/types/airdrop.go, Lines 11-56

Listing 4: (Line 20)

```
11 func (zd ZoneDrop) ValidateBasic() error {
12     errors := make(map[string]error)
13
14     // must be defined
15     if zd.ChainId == "" {
16         errors["ChainId"] = ErrUndefinedAttribute
17     }
18
19     // must be greater than 0
20     if zd.Duration.Microseconds() <= 0 {
21         errors["Duration"] = ErrInvalidDuration
22     }
23
24     // must be greater or equal to 0
25     // - equal will result in a full airdrop reward with immediate
   ↳  cut off on
26     //   expiery;
27     // - greater will result in a proportionally discounted
   ↳ airdrop reward over
28     //   the duration of decay;
29     if zd.Decay.Microseconds() < 0 {
30         errors["Decay"] = ErrInvalidDuration
31     }
32
33     // must be positive value
```

```go
    if zd.Allocation == 0 {
        errors["Allocation"] = ErrUndefinedAttribute
    }

    // must have at least one defined
    if zd.Actions == nil || len(zd.Actions) == 0 {
        errors["Actions"] = ErrUndefinedAttribute
    } else {
        wsum := sdk.ZeroDec()
        for _, aw := range zd.Actions {
            wsum = wsum.Add(aw)
        }
        if !wsum.Equal(sdk.OneDec()) {
            errors["Actions"] = fmt.Errorf("%w, got %s", ErrActionWeights, wsum)
        }
    }

    if len(errors) > 0 {
        return multierror.New(errors)
    }

    return nil
}
```

Risk Level:

Likelihood - 3
Impact - 4

Recommendation:

It is recommended that a check be placed on the upper/lower bound on the Duration parameter.

Remediation Plan:

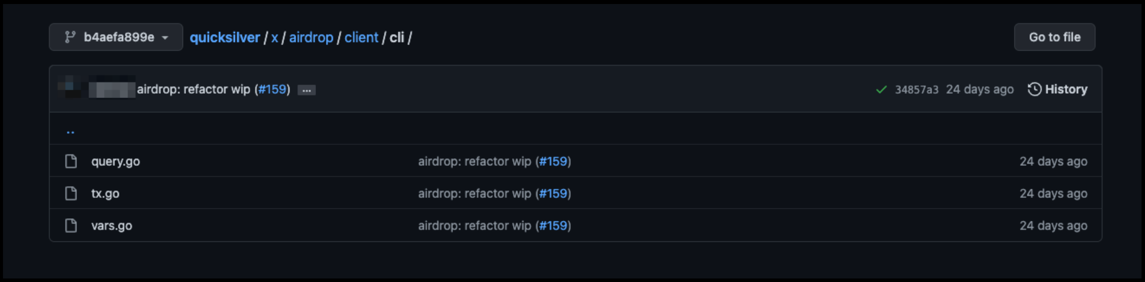
RISK ACCEPTED: The Quicksilver team accepted this risk of this finding. The Quicksilver team claim that Duration parameter will be decided by the governance.

3.5 (HAL-05) ONLY CLI IMPLEMENTED - LOW

Description:

It has been found that only the CLI has been implemented, with none of the REST tx or query methods enabled.

Location:



b4aefa899e quicksilver / x / airdrop / client / cli /

Go to file

airdrop: refactor wip (#159) 34857a3 24 days ago History

..		
query.go	airdrop: refactor wip (#159)	24 days ago
tx.go	airdrop: refactor wip (#159)	24 days ago
vars.go	airdrop: refactor wip (#159)	24 days ago

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

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

Remediation Plan:

RISK ACCEPTED: The Quicksilver team accepted this risk of this finding.

3.6 (HAL-06) UNHANDLED ERRORS - LOW

Description:

There are some instances where error handling has not been implemented for functions that might return an error.

Code Location:

x/airdrop/keeper/claim_record.go, Line 62

Listing 5

```
62      defer iterator.Close()
```

x/airdrop/keeper/claim_record.go, Line 85

Listing 6

```
85      defer iterator.Close()
```

x/airdrop/keeper/claim_record.go, Line 112

Listing 7

```
112     defer iterator.Close()
```

x/airdrop/keeper/proposal_handler.go, Line 84

Listing 8

```
84      defer iterator.Close()
```

x/airdrop/keeper/zonedrop.go, Line 60

Listing 9

```
defer iterator.Close()
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is recommended to implement the appropriate error checking to avoid unexpected crashes.

Remediation Plan:

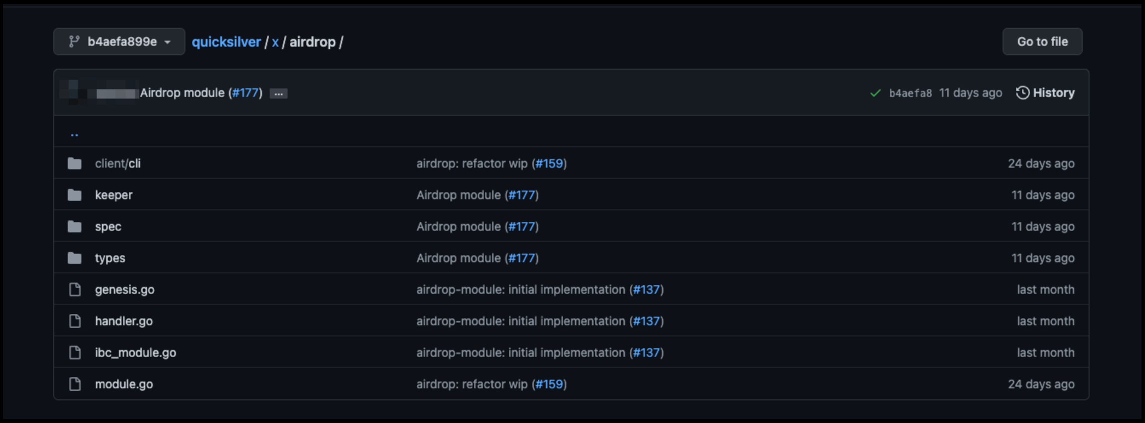
RISK ACCEPTED: The Quicksilver team accepted this risk of this finding.

3.7 (HAL-07) LACK OF SIMULATION AND FUZZING ON THE AIRDROP MODULE - LOW

Description:

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

Location:



b4aefa899e quicksilver / x / airdrop /   Go to file

[redacted] Airdrop module (#177) ...   b4aefa8 11 days ago   History

..		
client/cli	airdrop: refactor wip (#159)	24 days ago
keeper	Airdrop module (#177)	11 days ago
spec	Airdrop module (#177)	11 days ago
types	Airdrop module (#177)	11 days ago
genesis.go	airdrop-module: initial implementation (#137)	last month
handler.go	airdrop-module: initial implementation (#137)	last month
ibc_module.go	airdrop-module: initial implementation (#137)	last month
module.go	airdrop: refactor wip (#159)	24 days ago

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

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

- All module operations are included in the simulation module.
- The simulation uses a few accounts (e.g., between 5 and 20) to increase the likelihood of an interesting state change.
- The simulation uses the currencies/tokens that will be used in the production network.
- The simulation continues running when a transaction triggers an error.
- All transaction code paths are executed. (Enable code coverage to see how often individual lines are executed.)

Remediation Plan:

RISK ACCEPTED: The Quicksilver team accepted this risk of this finding.

3.8 (HAL-08) DUPLICATED ERROR CHECKS - INFORMATIONAL

Description:

There are two instances where an error check is not required, and the logic can be adjusted to only return the value.

Code Location:

x/airdrop/types/proposals.go, Lines 23-32

Listing 10: (Line 24)

```
23     err := govtypes.ValidateAbstract(m)
24     if err != nil {
25         return err
26     }
27
28     // validate ZoneDrop -> HandleRegisterZoneDropProposal
29
30     // validate ClaimRecords -> HandleRegisterZoneDropProposal (
   ↳ after decompression)
31
32     return nil
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Since the err variable will already be nil if the function has not generated any errors. An example of a piece of sufficient code:

Listing 11

```go
err := govtypes.ValidateAbstract(m)
return err
```

Remediation Plan:

SOLVED: The Quicksilver team solved the finding with the deleting duplicated error on this commit.

3.9 (HAL-09) ERRORS ARE RETURNED AS NIL - INFORMATIONAL

Description:

GetClaimableAmountForAction function returns the amount claimable for the given address. However, It returns as **nil** instead of more descriptive error.

Code Location:

x/airdrop/keeper/claim_record.go, Line 148

Listing 12

```
    if cr.Address == "" {
        return 0, nil
    }

    // action already completed, nothing to claim
    if _, exists := cr.ActionsCompleted[int32(action)]; exists {
        return 0, nil
    }

    // get zone airdrop details
    zd, ok := k.GetZoneDrop(ctx, chainID)
    if !ok {
        return 0, types.ErrZoneDropNotFound
    }

    // zone drop is not active, nothing to claim
    if !k.IsActiveZoneDrop(ctx, zd) {
        return 0, nil
    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Ensure that all error messages are compatible within the module. It is recommended to explain error messages more descriptive way.

Remediation Plan:

SOLVED: The Quicksilver team solved the finding with the returning more descriptive error on this commit.

3.10 (HAL-10) PANIC IS USED FOR ERROR HANDLING - INFORMATIONAL

Description:

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

Code Location:

Listing 13

```
./module.go:75:       panic(err)
./keeper/abci.go:15:             panic(err)
./keeper/keeper.go:40:       panic(fmt.Sprintf("%s module account
↳ has not been set", types.ModuleName))
./genesis.go:25:             panic(err)
./genesis.go:31:         panic("insufficient airdrop module account
↳  balance")
./genesis.go:39:             panic("zone sum not found")
./genesis.go:43:             panic("zone sum does not match zone
↳ allocation")
./genesis.go:59:             panic(err)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation Plan:

SOLVED: The Quicksilver team solved the finding with the returning more descriptive error on this commit.

3.11 (HAL-11) OPEN TODOs - INFORMATIONAL

Description:

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

Code Location:

Listing 14

```
./keeper/claim_action.go:16:// TODO: we also want to verify that
↳ the action was executed on the remote
./keeper/claim_action.go:44:        // TODO: implement check once
↳ GbP is implemented
./keeper/claim_action.go:47:        // TODO: implement
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

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

Remediation Plan:

SOLVED: The Quicksilver team solved the finding with the deleting file on this commit.

AUTOMATED TESTING

Description:

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

Semgrep - Security Analysis Output Sample:

Listing 15: Rule Set

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

Semgrep Results:

Listing 16

```
Findings:

  types/proposals.go
     dgryski.semgrep-go.errnilcheck.err-nil-check
```

```
        superfluous nil err check before return
        Details: https://sg.run/5Qd6

         24 if err != nil {
         25     return err
         26 }
         27
         28 // validate ZoneDrop -> HandleRegisterZoneDropProposal
         29
         30 // validate ClaimRecords ->
↳ HandleRegisterZoneDropProposal (after decompression)
         31
         32 return nil


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

         70 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
         97 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
        124 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
        151 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
        256 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
        267 protoReq.Address, err = runtime.String(val)
          ----------------------------------------
        294 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
        305 protoReq.Address, err = runtime.String(val)
          ----------------------------------------
        336 protoReq.ChainId, err = runtime.String(val)
          ----------------------------------------
        370 protoReq.ChainId, err = runtime.String(val)
```

THANK YOU FOR CHOOSING

// HALBORN