Haqq - Coinomics Module

Cosmos Security Assessment

Prepared by: **Halborn**

Date of Engagement: **November 27th, 2023 - December 15th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 3
CONTACTS 3
1 EXECUTIVE OVERVIEW 4
1.1 INTRODUCTION 5
1.2 ASSESSMENT SUMMARY 6
1.3 SCOPE 7
2 RISK METHODOLOGY 8
2.1 EXPLOITABILITY 9
2.2 IMPACT 10
2.3 SEVERITY COEFFICIENT 12
2.4 TEST APPROACH & METHODOLOGY 14
3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 15
4 FINDINGS & TECH DETAILS 16
4.1 (HAL-01) USE OF VULNERABLE DEPENDENCIES - LOW(4.4) 18
Description 18
Code Location 18
BVSS 18
Recommendation 18
Remediation Plan 19
5 MANUAL TESTING 20
5.1 SCENARIOS TESTED 21
RESULTS 22
6 AUTOMATED TESTING 24
6.1 Description 25

6.2 Semgrep 25
Security Analysis Output Sample 25
Semgrep Results 26
6.3 Gosec 26
Analysis Output Sample 26
6.4 StaticCheck 26
Analysis Output Sample 26
6.5 CodeQL 27
Analysis Output Sample (go and cosmos queries) 27
6.6 Nancy 28
Analysis Output Sample 28

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	12/14/2023
0.2	Document Updates	12/15/2023
0.3	Draft Review	12/15/2023
0.4	Draft Review	12/15/2023
1.0	Remediation Plan	12/19/2023
1.1	Remediation Plan Review	12/19/2023
1.2	Remediation Plan Review	12/19/2023

CONTACTS

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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Haqq engaged Halborn to conduct a security assessment on their app chain module beginning on November 27th, 2023 and ending on December 15th, 2023. The security assessment was scoped to the coinomics module provided to the Halborn team.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned one full-time security engineer to assessment the security of the merge requests. 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 assessment is to:

- Ensure that the **Cosmos Module** operates as intended.
- Identify potential security issues with the coinomics module.

In summary, Halborn identified one low issue that was successfully addressed by the Haqq team.

1.3 SCOPE

IN-SCOPE CODE & COMMIT:

- Repository: haqq-network/haqq
  - Commit ID: 9e61518ead7ceef38193970c3a311133c0421de8
  - Module **in scope**:
    - x/coinomics.

REMEDIATION COMMIT IDs:

- 38679c5d2ad3cf7ff4c5edad3189a154626cd25b

2. 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.

2.1 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:

$$E = \prod m_e$$

2.2 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)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

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

2.3 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.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	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
Low	2 - 4.4
Informational	0 - 1.9

2.4 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 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 codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis on files and modules in-scope.

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	1	0

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) USE OF VULNERABLE DEPENDENCIES	Low (4.4)	SOLVED - 12/19/2023

FINDINGS & TECH DETAILS

4.1 (HAL-01) USE OF VULNERABLE DEPENDENCIES - LOW (4.4)

Description:

A variety of vulnerabilities exists in dependencies used by the project’s coinomics module.

Code Location:

Vulnerabilities flagged by the tool nancy:

ID	Package	Rating	Description
CVE-2022-44797	btcd	CRITICAL	Reachable Assertion
CVE-2022-39389	btcd	MEDIUM	Improper Input Validation

ID	Package	Rating	Description
CVE-2023-28642	runc	HIGH	Improper Preservation of Permissions
CVE-2023-27561	runc	HIGH	Incorrectly-Resolved Name or Reference
CVE-2023-25809	runc	MEDIUM	Improper Preservation of Permissions

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:L/D:L/Y:L/R:N/S:U (4.4)

Recommendation:

Where possible, keep dependencies patched in order to reduce the risk of the system being attacked using known vulnerabilities. It is recommended that the Haqq team runs the nancy and govulncheck, tools regularly and fix as many warnings as possible.

Remediation Plan:

SOLVED: The Haqq team solved the issue by upgrading the btcd dependency to a new version.

Commit ID: 38679c5d2ad3cf7ff4c5edad3189a154626cd25b

MANUAL TESTING

5.1 SCENARIOS TESTED

In the manual testing phase, the following scenarios were simulated. It must be taken into account that it's been reviewed every test found in x/coinomics and sub-folders, which almost covers each component of the project. The following scenarios are covered in these unit tests:

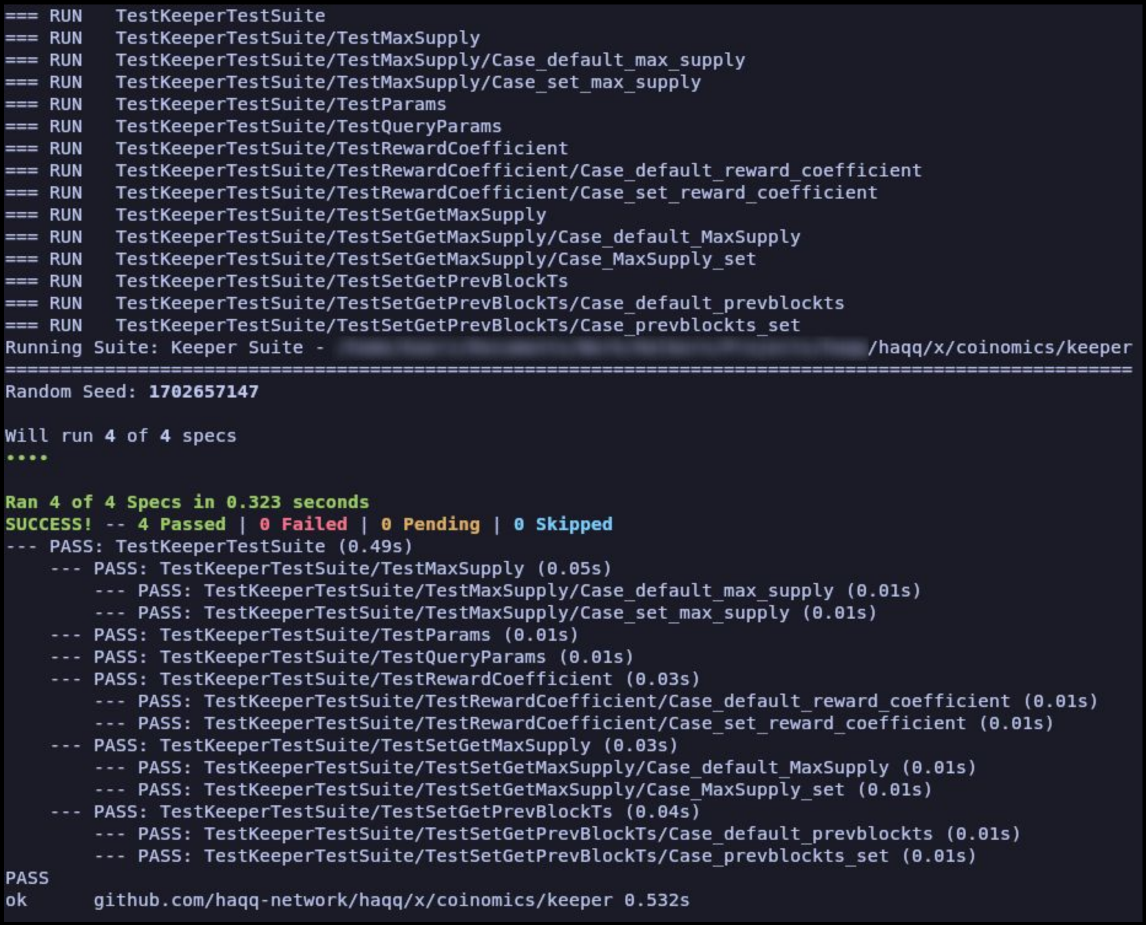
- Tests 1: Keeper tests
  - Tests 1.1: Tests for MaxSupply feature
    - Test 1.1.1: Test using MaxSupply default value.
    - Test 1.1.2: Test using MaxSupply value set in run-time.
  - Test 1.2: Tests for module params.
  - Test 1.3: Tests for module query params
  - Tests 1.4: Tests for RewardCoefficient feature
    - Test 1.4.1: Test using RewardCoefficient default value.
    - Test 1.4.2: Test using RewardCoefficient value set in run-time.
  - Tests 1.5: Tests for get/set MaxSupply methods
    - Test 1.5.1: Test using get/set methods with MaxSupply default value.
    - Test 1.5.2: Test using get/set methods with MaxSupply value set in run-time.
  - Tests 1.6: Tests for get/set PrevBlockTs methods
    - Test 1.6.1: Test using get/set methods with PrevBlockTs default value.
    - Test 1.6.2: Test using get/set methods with PrevBlockTs value set in run-time.
- Tests 2: Genesis tests
  - Test 2.1: Genesis validation tests

The general module executing was tested using ginkgo, simulating some parts of the code and states that the module will reach during its execution.

- Simulations 1: Mint results when coinomics module is disables.
- Simulations 2: Mint results when coinomics is enabled on a regular year.
- Simulations 3: Mint results when coinomics is enabled on a leap year.
- Simulations 4: Mint results when coinomics is enabled and MaxSupply is reached.

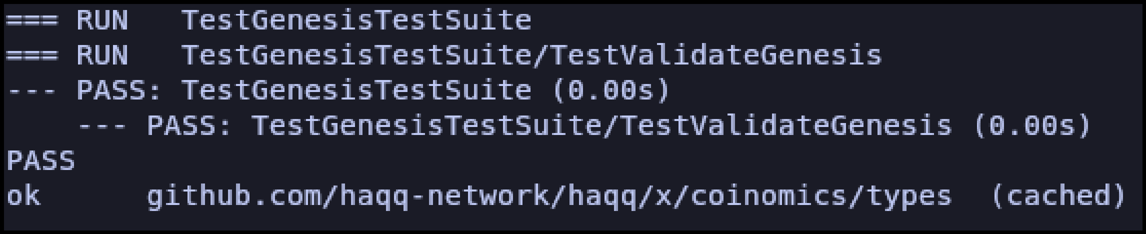
These tests and simulations cover an 85.2% of the coinomics module.

RESULTS:



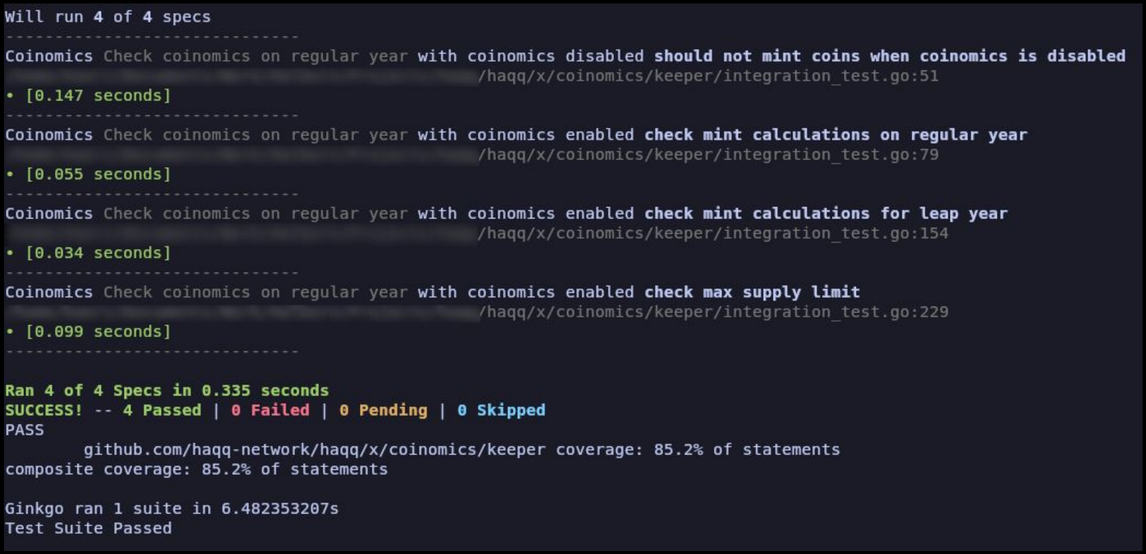
```
=== RUN   TestKeeperTestSuite
=== RUN   TestKeeperTestSuite/TestMaxSupply
=== RUN   TestKeeperTestSuite/TestMaxSupply/Case_default_max_supply
=== RUN   TestKeeperTestSuite/TestMaxSupply/Case_set_max_supply
=== RUN   TestKeeperTestSuite/TestParams
=== RUN   TestKeeperTestSuite/TestQueryParams
=== RUN   TestKeeperTestSuite/TestRewardCoefficient
=== RUN   TestKeeperTestSuite/TestRewardCoefficient/Case_default_reward_coefficient
=== RUN   TestKeeperTestSuite/TestRewardCoefficient/Case_set_reward_coefficient
=== RUN   TestKeeperTestSuite/TestSetGetMaxSupply
=== RUN   TestKeeperTestSuite/TestSetGetMaxSupply/Case_default_MaxSupply
=== RUN   TestKeeperTestSuite/TestSetGetMaxSupply/Case_MaxSupply_set
=== RUN   TestKeeperTestSuite/TestSetGetPrevBlockTs
=== RUN   TestKeeperTestSuite/TestSetGetPrevBlockTs/Case_default_prevblockts
=== RUN   TestKeeperTestSuite/TestSetGetPrevBlockTs/Case_prevblockts_set
Running Suite: Keeper Suite - [redacted]/haqq/x/coinomics/keeper
====================================================================================================
Random Seed: 1702657147

Will run 4 of 4 specs
••••

Ran 4 of 4 Specs in 0.323 seconds
SUCCESS! -- 4 Passed | 0 Failed | 0 Pending | 0 Skipped
--- PASS: TestKeeperTestSuite (0.49s)
    --- PASS: TestKeeperTestSuite/TestMaxSupply (0.05s)
        --- PASS: TestKeeperTestSuite/TestMaxSupply/Case_default_max_supply (0.01s)
        --- PASS: TestKeeperTestSuite/TestMaxSupply/Case_set_max_supply (0.01s)
    --- PASS: TestKeeperTestSuite/TestParams (0.01s)
    --- PASS: TestKeeperTestSuite/TestQueryParams (0.01s)
    --- PASS: TestKeeperTestSuite/TestRewardCoefficient (0.03s)
        --- PASS: TestKeeperTestSuite/TestRewardCoefficient/Case_default_reward_coefficient (0.01s)
        --- PASS: TestKeeperTestSuite/TestRewardCoefficient/Case_set_reward_coefficient (0.01s)
    --- PASS: TestKeeperTestSuite/TestSetGetMaxSupply (0.03s)
        --- PASS: TestKeeperTestSuite/TestSetGetMaxSupply/Case_default_MaxSupply (0.01s)
        --- PASS: TestKeeperTestSuite/TestSetGetMaxSupply/Case_MaxSupply_set (0.01s)
    --- PASS: TestKeeperTestSuite/TestSetGetPrevBlockTs (0.04s)
        --- PASS: TestKeeperTestSuite/TestSetGetPrevBlockTs/Case_default_prevblockts (0.01s)
        --- PASS: TestKeeperTestSuite/TestSetGetPrevBlockTs/Case_prevblockts_set (0.01s)
PASS
ok      github.com/haqq-network/haqq/x/coinomics/keeper 0.532s
```

Figure 1: Keeper unit tests results



```
=== RUN   TestGenesisTestSuite
=== RUN   TestGenesisTestSuite/TestValidateGenesis
--- PASS: TestGenesisTestSuite (0.00s)
    --- PASS: TestGenesisTestSuite/TestValidateGenesis (0.00s)
PASS
ok      github.com/haqq-network/haqq/x/coinomics/types  (cached)
```

Figure 2: Genesis unit tests results



```
Will run 4 of 4 specs
------------------------------
Coinomics Check coinomics on regular year with coinomics disabled should not mint coins when coinomics is disabled
[redacted]/haqq/x/coinomics/keeper/integration_test.go:51
• [0.147 seconds]
------------------------------
Coinomics Check coinomics on regular year with coinomics enabled check mint calculations on regular year
[redacted]/haqq/x/coinomics/keeper/integration_test.go:79
• [0.055 seconds]
------------------------------
Coinomics Check coinomics on regular year with coinomics enabled check mint calculations for leap year
[redacted]/haqq/x/coinomics/keeper/integration_test.go:154
• [0.034 seconds]
------------------------------
Coinomics Check coinomics on regular year with coinomics enabled check max supply limit
[redacted]/haqq/x/coinomics/keeper/integration_test.go:229
• [0.099 seconds]
------------------------------

Ran 4 of 4 Specs in 0.335 seconds
SUCCESS! -- 4 Passed | 0 Failed | 0 Pending | 0 Skipped
PASS
        github.com/haqq-network/haqq/x/coinomics/keeper coverage: 85.2% of statements
composite coverage: 85.2% of statements

Ginkgo ran 1 suite in 6.482353207s
Test Suite Passed
```

Figure 3: Ginkgo tests results

AUTOMATED TESTING

6.1 Description

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec, semgrep, codeQL 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.

6.2 Semgrep

Security Analysis Output Sample:

Listing 1: Rule Set

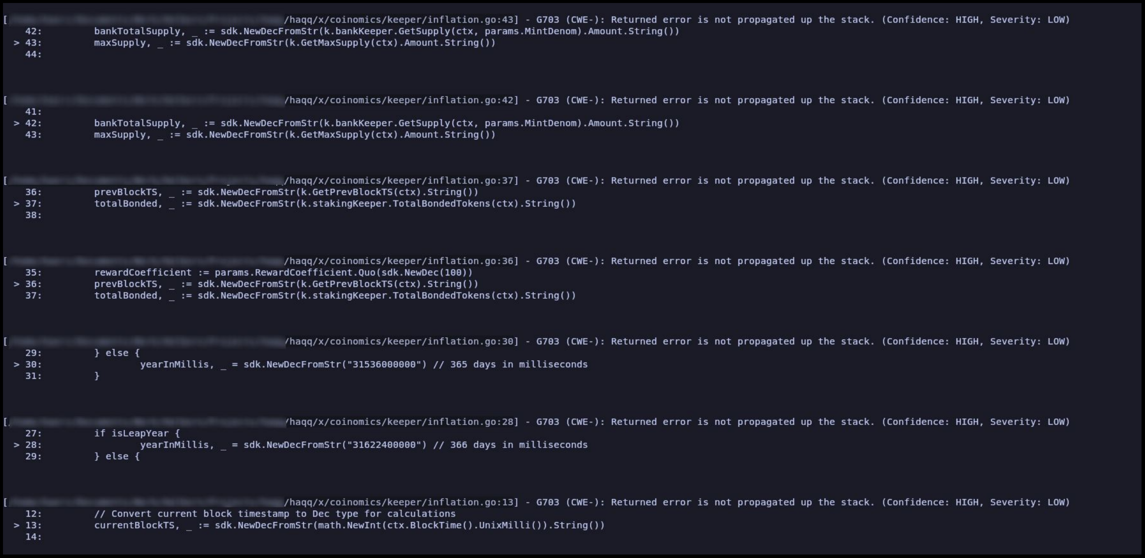
```
semgrep --config "p/dgryski.semgrep-go" x/liquidstakeibc --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o dgryski.semgrep
semgrep --config "p/owasp-top-ten"       x/liquidstakeibc --exclude='*_test.go' --max-lines-per-finding 1000 --no-git-ignore -o owasp-top-ten.semgrep
semgrep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

Semgrep Results:

- No major issues found by Semgrep.

6.3 Gosec

Analysis Output Sample:



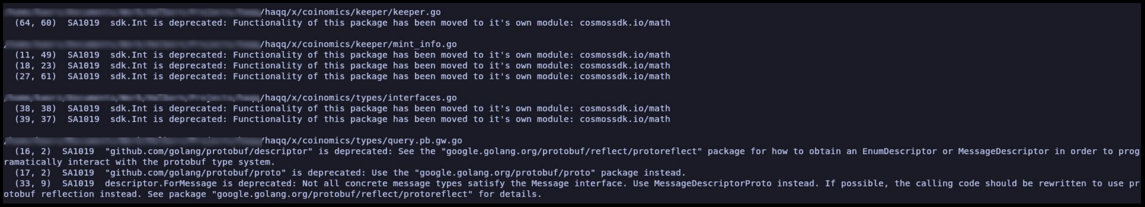
```
[[redacted]/haqq/x/coinomics/keeper/inflation.go:43] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    42:         bankTotalSupply, _ := sdk.NewDecFromStr(k.bankKeeper.GetSupply(ctx, params.MintDenom).Amount.String())
  > 43:         maxSupply, _ := sdk.NewDecFromStr(k.GetMaxSupply(ctx).Amount.String())
    44:



[[redacted]/haqq/x/coinomics/keeper/inflation.go:42] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    41:
  > 42:         bankTotalSupply, _ := sdk.NewDecFromStr(k.bankKeeper.GetSupply(ctx, params.MintDenom).Amount.String())
    43:         maxSupply, _ := sdk.NewDecFromStr(k.GetMaxSupply(ctx).Amount.String())



[[redacted]/haqq/x/coinomics/keeper/inflation.go:37] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    36:         prevBlockTS, _ := sdk.NewDecFromStr(k.GetPrevBlockTS(ctx).String())
  > 37:         totalBonded, _ := sdk.NewDecFromStr(k.stakingKeeper.TotalBondedTokens(ctx).String())
    38:



[[redacted]/haqq/x/coinomics/keeper/inflation.go:36] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    35:         rewardCoefficient := params.RewardCoefficient.Quo(sdk.NewDec(100))
  > 36:         prevBlockTS, _ := sdk.NewDecFromStr(k.GetPrevBlockTS(ctx).String())
    37:         totalBonded, _ := sdk.NewDecFromStr(k.stakingKeeper.TotalBondedTokens(ctx).String())



[[redacted]/haqq/x/coinomics/keeper/inflation.go:30] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    29:         } else {
  > 30:                 yearInMillis, _ = sdk.NewDecFromStr("31536000000") // 365 days in milliseconds
    31:         }



[[redacted]/haqq/x/coinomics/keeper/inflation.go:28] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    27:         if isLeapYear {
  > 28:                 yearInMillis, _ = sdk.NewDecFromStr("31622400000") // 366 days in milliseconds
    29:         } else {



[[redacted]/haqq/x/coinomics/keeper/inflation.go:13] - G703 (CWE-): Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    12:         // Convert current block timestamp to Dec type for calculations
  > 13:         currentBlockTS, _ := sdk.NewDecFromStr(math.NewInt(ctx.BlockTime().UnixMilli()).String())
    14:
```

Figure 4: Gosec results

- No major issues found by Gosec.

6.4 StaticCheck

Analysis Output Sample:



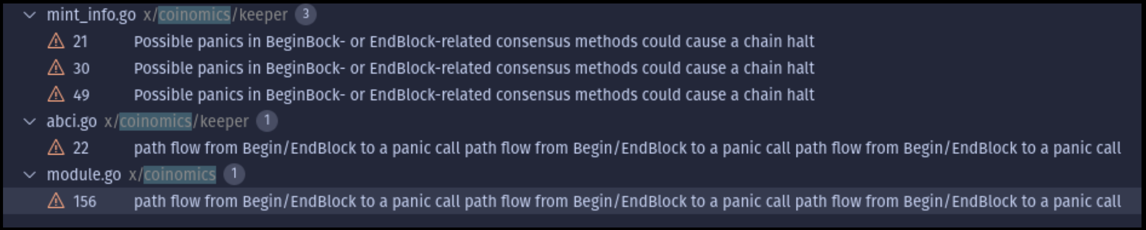
```
[redacted]/haqq/x/coinomics/keeper/keeper.go
  (64, 60)  SA1019  sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math

[redacted]/haqq/x/coinomics/keeper/mint_info.go
  (11, 49)  SA1019  sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math
  (18, 23)  SA1019  sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math
  (27, 61)  SA1019  sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math

[redacted]/haqq/x/coinomics/types/interfaces.go
  (38, 38)  SA1019  sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math
  (39, 37)  SA1019  sdk.Int is deprecated: Functionality of this package has been moved to it's own module: cosmossdk.io/math

[redacted]/haqq/x/coinomics/types/query.pb.gw.go
  (16, 2)  SA1019  "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 prog
ramatically interact with the protobuf type system.
  (17, 2)  SA1019  "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.org/protobuf/proto" package instead.
  (33, 9)  SA1019  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 pr
otobuf reflection instead. See package "google.golang.org/protobuf/reflect/protoreflect" for details.
```

Figure 5: StaticCheck results

- No major issues found by StaticCheck.

6.5 CodeQL

Analysis Output Sample (go and cosmos queries):



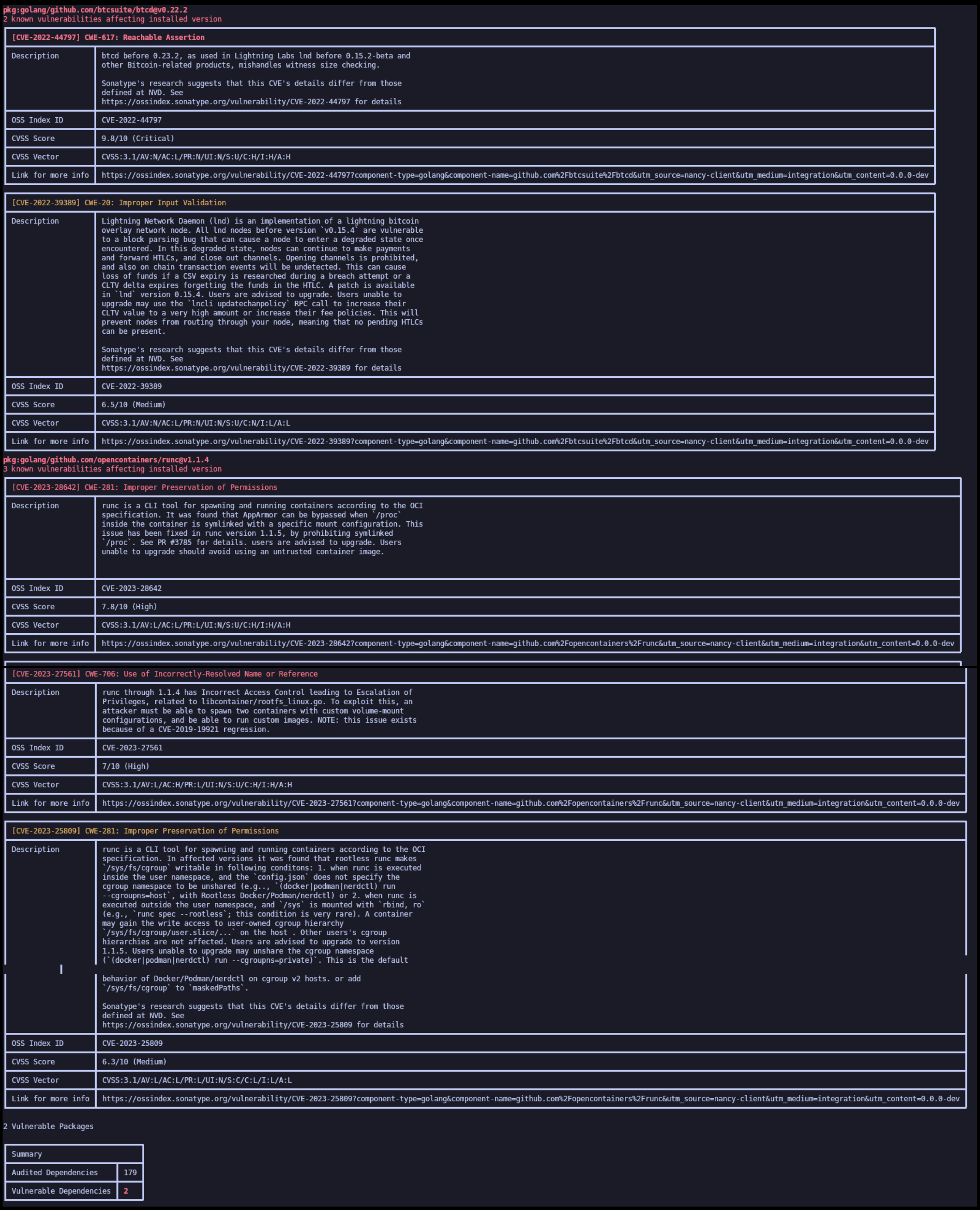
```
⌄ mint_info.go x/coinomics/keeper 3
  ⚠ 21    Possible panics in BeginBock- or EndBlock-related consensus methods could cause a chain halt
  ⚠ 30    Possible panics in BeginBock- or EndBlock-related consensus methods could cause a chain halt
  ⚠ 49    Possible panics in BeginBock- or EndBlock-related consensus methods could cause a chain halt
⌄ abci.go x/coinomics/keeper 1
  ⚠ 22    path flow from Begin/EndBlock to a panic call path flow from Begin/EndBlock to a panic call path flow from Begin/EndBlock to a panic call
⌄ module.go x/coinomics 1
  ⚠ 156   path flow from Begin/EndBlock to a panic call path flow from Begin/EndBlock to a panic call path flow from Begin/EndBlock to a panic call
```

Figure 6: CodeQL results

- No major issues found by CodeQL in scoped module.

6.6 Nancy

Analysis Output Sample:



```
pkg:golang/github.com/btcsuite/btcd@v0.22.2
2 known vulnerabilities affecting installed version
┌──────────────────────────────────────────────────────────────────────────────┐
│ [CVE-2022-44797] CWE-617: Reachable Assertion                                │
├──────────────────┬───────────────────────────────────────────────────────────┤
│ Description      │ btcd before 0.23.2, as used in Lightning Labs lnd before 0.15.2-beta and
│                  │ other Bitcoin-related products, mishandles witness size checking.
│                  │
│                  │ Sonatype's research suggests that this CVE's details differ from those
│                  │ defined at NVD. See
│                  │ https://ossindex.sonatype.org/vulnerability/CVE-2022-44797 for details
├──────────────────┼───────────────────────────────────────────────────────────┤
│ OSS Index ID     │ CVE-2022-44797                                            │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Score       │ 9.8/10 (Critical)                                         │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Vector      │ CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H              │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ Link for more info │ https://ossindex.sonatype.org/vulnerability/CVE-2022-44797?component-type=golang&component-name=github.com%2Fbtcsuite%2Fbtcd&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev │
└──────────────────┴───────────────────────────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
│ [CVE-2022-39389] CWE-20: Improper Input Validation                           │
├──────────────────┬───────────────────────────────────────────────────────────┤
│ Description      │ Lightning Network Daemon (lnd) is an implementation of a lightning bitcoin
│                  │ overlay network node. All lnd nodes before version `v0.15.4` are vulnerable
│                  │ to a block parsing bug that can cause a node to enter a degraded state once
│                  │ encountered. In this degraded state, nodes can continue to make payments
│                  │ and forward HTLCs, and close out channels. Opening channels is prohibited,
│                  │ and also on chain transaction events will be undetected. This can cause
│                  │ loss of funds if a CSV expiry is researched during a breach attempt or a
│                  │ CLTV delta expires forgetting the funds in the HTLC. A patch is available
│                  │ in `lnd` version 0.15.4. Users are advised to upgrade. Users unable to
│                  │ upgrade may use the `lncli updatechanpolicy` RPC call to increase their
│                  │ CLTV value to a very high amount or increase their fee policies. This will
│                  │ prevent nodes from routing through your node, meaning that no pending HTLCs
│                  │ can be present.
│                  │
│                  │ Sonatype's research suggests that this CVE's details differ from those
│                  │ defined at NVD. See
│                  │ https://ossindex.sonatype.org/vulnerability/CVE-2022-39389 for details
├──────────────────┼───────────────────────────────────────────────────────────┤
│ OSS Index ID     │ CVE-2022-39389                                            │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Score       │ 6.5/10 (Medium)                                           │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Vector      │ CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L              │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ Link for more info │ https://ossindex.sonatype.org/vulnerability/CVE-2022-39389?component-type=golang&component-name=github.com%2Fbtcsuite%2Fbtcd&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev │
└──────────────────┴───────────────────────────────────────────────────────────┘
pkg:golang/github.com/opencontainers/runc@v1.1.4
3 known vulnerabilities affecting installed version
┌──────────────────────────────────────────────────────────────────────────────┐
│ [CVE-2023-28642] CWE-281: Improper Preservation of Permissions               │
├──────────────────┬───────────────────────────────────────────────────────────┤
│ Description      │ runc is a CLI tool for spawning and running containers according to the OCI
│                  │ specification. It was found that AppArmor can be bypassed when `/proc`
│                  │ inside the container is symlinked with a specific mount configuration. This
│                  │ issue has been fixed in runc version 1.1.5, by prohibiting symlinked
│                  │ `/proc`. See PR #3785 for details. users are advised to upgrade. Users
│                  │ unable to upgrade should avoid using an untrusted container image.
│                  │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ OSS Index ID     │ CVE-2023-28642                                            │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Score       │ 7.8/10 (High)                                             │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Vector      │ CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H              │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ Link for more info │ https://ossindex.sonatype.org/vulnerability/CVE-2023-28642?component-type=golang&component-name=github.com%2Fopencontainers%2Frunc&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev │
└──────────────────┴───────────────────────────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
│ [CVE-2023-27561] CWE-706: Use of Incorrectly-Resolved Name or Reference      │
├──────────────────┬───────────────────────────────────────────────────────────┤
│ Description      │ runc through 1.1.4 has Incorrect Access Control leading to Escalation of
│                  │ Privileges, related to libcontainer/rootfs_linux.go. To exploit this, an
│                  │ attacker must be able to spawn two containers with custom volume-mount
│                  │ configurations, and be able to run custom images. NOTE: this issue exists
│                  │ because of a CVE-2019-19921 regression.
├──────────────────┼───────────────────────────────────────────────────────────┤
│ OSS Index ID     │ CVE-2023-27561                                            │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Score       │ 7/10 (High)                                               │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Vector      │ CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H              │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ Link for more info │ https://ossindex.sonatype.org/vulnerability/CVE-2023-27561?component-type=golang&component-name=github.com%2Fopencontainers%2Frunc&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev │
└──────────────────┴───────────────────────────────────────────────────────────┘
┌──────────────────────────────────────────────────────────────────────────────┐
│ [CVE-2023-25809] CWE-281: Improper Preservation of Permissions               │
├──────────────────┬───────────────────────────────────────────────────────────┤
│ Description      │ runc is a CLI tool for spawning and running containers according to the OCI
│                  │ specification. In affected versions it was found that rootless runc makes
│                  │ `/sys/fs/cgroup` writable in following conditons: 1. when runc is executed
│                  │ inside the user namespace, and the `config.json` does not specify the
│                  │ cgroup namespace to be unshared (e.g.., `(docker|podman|nerdctl) run
│                  │ --cgroupns=host`, with Rootless Docker/Podman/nerdctl) or 2. when runc is
│                  │ executed outside the user namespace, and `/sys` is mounted with `rbind, ro`
│                  │ (e.g., `runc spec --rootless`; this condition is very rare). A container
│                  │ may gain the write access to user-owned cgroup hierarchy
│                  │ `/sys/fs/cgroup/user.slice/...` on the host . Other users's cgroup
│                  │ hierarchies are not affected. Users are advised to upgrade to version
│                  │ 1.1.5. Users unable to upgrade may unshare the cgroup namespace
│                  │ (`(docker|podman|nerdctl) run --cgroupns=private)`. This is the default
         |
│                  │ behavior of Docker/Podman/nerdctl on cgroup v2 hosts. or add
│                  │ `/sys/fs/cgroup` to `maskedPaths`.
│                  │
│                  │ Sonatype's research suggests that this CVE's details differ from those
│                  │ defined at NVD. See
│                  │ https://ossindex.sonatype.org/vulnerability/CVE-2023-25809 for details
├──────────────────┼───────────────────────────────────────────────────────────┤
│ OSS Index ID     │ CVE-2023-25809                                            │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Score       │ 6.3/10 (Medium)                                           │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ CVSS Vector      │ CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:L/I:L/A:L              │
├──────────────────┼───────────────────────────────────────────────────────────┤
│ Link for more info │ https://ossindex.sonatype.org/vulnerability/CVE-2023-25809?component-type=golang&component-name=github.com%2Fopencontainers%2Frunc&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-dev │
└──────────────────┴───────────────────────────────────────────────────────────┘

2 Vulnerable Packages

┌───────────────────────────┐
│ Summary                   │
├───────────────────────┬───┤
│ Audited Dependencies  │179│
├───────────────────────┼───┤
│ Vulnerable Dependencies │ 2 │
└───────────────────────┴───┘
```

- No major issues found by Nancy.

THANK YOU FOR CHOOSING

HALBORN