Maya Protocol - Liquidity Auction

Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 15th, 2022 - November 20th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/17/2022	John Saigle
0.2	Document Updates	11/20/2022	John Saigle
0.3	Draft Review	11/22/2022	Gabi Urrutia
1.0	Remediation Plan	01/06/2023	John Saigle
1.1	Remediation Plan Review	01/06/2023	Gokberk Gulgun
1.2	Remediation Plan Review	01/09/2023	Gabi Urrutia
1.3	Remediation Plan Update	01/17/2023	John Saigle
1.4	Remediation Plan Review	01/18/2023	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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Maya Protocol engaged Halborn to conduct a security audit on their liquidity auction module, beginning on November 15th, 2022 and ending on November 20th, 2022. The security assessment was scoped to the sections of code that pertain to the Add Liquidity feature.

1.2 AUDIT SUMMARY

The team at Halborn was provided one week for the engagement and assigned a full-time security engineer to audit the security of the liquidity auction feature. The security engineer is a blockchain and smart contract security expert with advanced penetration testing, smart contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that the new Liquidity Auction feature operates as intended.
- Identify potential security issues with the custom modules used in Maya.

In summary, Halborn identified security risks that could have a negative impact on the protocol. The Maya team mostly addressed these issues.

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 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 audit:

- 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 related to the Add Liquidity feature.

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

1. IN-SCOPE TREE & COMMIT :

This review was scoped to the Maya thornode repository, branch MAYA-70.

- URL: https://gitlab.com/mayachain/thornode/-/blob/MAYA-70
- Commit hash: 6a33c208bd2d6d54113b4bea792bb75b6c90112f

Special attention was paid to the following sections of code which were supplied by the Maya development team. They have communicated that these areas cover the addition of the Add Liquidity feature.

- x/thorchain/helpers.go, Lines 713-729
- x/thorchain/handler_donate.go, Lines 81-99
- x/thorchain/handler_donate.go, the addLiquidityFromDonate function

Note that Halborn has conducted a previous audit of the Maya codebase and that some points raised during that review are still outstanding. This report covers only the Add Liquidity features, and the findings reported below supplement the earlier report. All security-related findings in both reports should be addressed.

2. REMEDIATION PRs & COMMITS:

- https://gitlab.com/mayachain/thornode/-/merge_requests/19
- https://gitlab.com/mayachain/thornode/-/merge_requests/23
- https://gitlab.com/mayachain/thornode/-/merge_requests/24
- https://gitlab.com/mayachain/thornode/-/merge_requests/25

Commit IDS :

- f37b82b3f486c5e387b51e09fce733d8582124ac

- f59a8d61f1db443bdc0b938d451e1b0375fe7f03
- 65014b986b9776626c217c27354cc7ae02227a21
- 27439445d2429723203a27581d9a70eedd1e7db8

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	2	1	4	2

LIKELIHOOD

IMPACT

		(HAL-01) (HAL-02)		
(HAL-04)		(HAL-03)		
	(HAL-05) (HAL-06) (HAL-07)			
(HAL-08) (HAL-09)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - UNAVAILABLE POOLS CAN BE MODIFIED BY LIQUIDITY DONATIONS	High	SOLVED - 12/28/2022
HAL-02 - LIQUIDITY PROVIDERS MAY LOSE FUNDS WHEN DONATING TO UNAVAILABLE POOLS	High	SOLVED - 12/28/2022
HAL-03 - USE OF CRITICALLY VULNERABLE COSMOSSDK VERSION	Medium	SOLVED - 01/06/2023
HAL-04 - USE OF UNSUPPORTED GO VERSION	Low	SOLVED - 01/06/2023
HAL-05 - LACK OF UNIT TEST FOR LIQUIDITY POOL AFTER LIQUIDITY DONATION	Low	SOLVED - 01/06/2023
HAL-06 - MNEMONIC PHRASES ARE EXPOSED THROUGH THE REPOSITORY	Low	RISK ACCEPTED
HAL-07 - UNCHECKED ERRORS	Low	SOLVED - 01/06/2023
HAL-08 - MISLEADING FUNCTION NAME	Informational	SOLVED - 01/06/2023
HAL-09 - SPELLING MISTAKES	Informational	SOLVED - 01/06/2023

FINDINGS & TECH DETAILS

3.1 (HAL-01) UNAVAILABLE POOLS CAN BE MODIFIED BY LIQUIDITY DONATIONS - HIGH

Description:

When a Liquidity Provider sends a Message containing an action that donates liquidity, it is possible to donate to a Pool that is not available. This can have unintended consequences, including modifications of suspended pools.

Code Location:

In the addLiquidityFromDonate function, there is a call to the SetPool function before the status of the Pool is checked. The SetPool modifies the data within the pool.

x/thorchain/handler_donate.go, Lines 116-208

Listing 1: SetPool is called before the Pool.Status is checked. (Lines 150,161)

```
116 func (h DonateHandler) addLiquidityFromDonate(ctx cosmos.Context,
 ↳ msg MsgDonate, pool types.Pool, su types.LiquidityProvider) error
 ↳ {
117     ver := h.mgr.Keeper().GetLowestActiveVersion(ctx)
118     synthSupply := h.mgr.Keeper().GetTotalSupply(ctx, pool.Asset.
 ↳ GetSyntheticAsset())
119     originalUnits := pool.CalcUnits(ver, synthSupply)
120
121     pendingAssetAmt := su.PendingAsset
122     pendingRuneAmt := common.GetSafeShare(pendingAssetAmt, pool.
 ↳ PendingInboundAsset, msg.RuneAmount)
123
124     pool.PendingInboundRune = common.SafeSub(pool.
 ↳ PendingInboundRune, su.PendingRune)
125     pool.PendingInboundAsset = common.SafeSub(pool.
 ↳ PendingInboundAsset, su.PendingAsset)
```

```
126     su.PendingAsset = cosmos.ZeroUint()
127     su.PendingRune = cosmos.ZeroUint()
128     su.PendingTxID = ""
129 
130     ctx.Logger().Info("pre add liquidity", "pool", pool.Asset, "
 ↳ rune", pool.BalanceRune, "asset", pool.BalanceAsset, "LP units",
 ↳ pool.LPUnits, "synth units", pool.SynthUnits)
131 
132     balanceRune := pool.BalanceRune
133     balanceAsset := pool.BalanceAsset
134 
135     oldPoolUnits := pool.GetPoolUnits()
136     newPoolUnits, liquidityUnits, err := calculatePoolUnitsV1(
 ↳ oldPoolUnits, balanceRune, balanceAsset, pendingRuneAmt,
 ↳ pendingAssetAmt)
137     if err != nil {
138         return ErrInternal(err, "fail to calculate pool unit")
139     }
140     ctx.Logger().Info("current pool status", "pool units",
 ↳ newPoolUnits, "liquidity units", liquidityUnits)
141     poolRune := balanceRune.Add(pendingRuneAmt)
142     poolAsset := balanceAsset.Add(pendingAssetAmt)
143     pool.LPUnits = pool.LPUnits.Add(liquidityUnits)
144     pool.BalanceRune = poolRune
145     pool.BalanceAsset = poolAsset
146     ctx.Logger().Info("post add liquidity", "pool", pool.Asset, "
 ↳ rune", pool.BalanceRune, "asset", pool.BalanceAsset, "LP units",
 ↳ pool.LPUnits, "synth units", pool.SynthUnits, "add liquidity units
 ↳ ", liquidityUnits)
147     if pool.BalanceRune.IsZero() || pool.BalanceAsset.IsZero() {
148         return ErrInternal(err, "pool cannot have zero rune or
 ↳ asset balance")
149     }
150     if err := h.mgr.Keeper().SetPool(ctx, pool); err != nil {
151         return ErrInternal(err, "fail to save pool")
152     }
153     if originalUnits.IsZero() && !pool.GetPoolUnits().IsZero() {
154         poolEvent := NewEventPool(pool.Asset, pool.Status)
155         if err := h.mgr.EventMgr().EmitEvent(ctx, poolEvent); err
 ↳ != nil {
156             ctx.Logger().Error("fail to emit pool event", "error",
 ↳  err)
157         }
158     }
```

```
159
160     su.Units = su.Units.Add(liquidityUnits)
161     if pool.Status == PoolAvailable {
```

Recommendation:

Refactor the code so that the Pool is updated only after all relevant values have been verified. Review the design of the protocol to determine the consequences of donating to unavailable pools.

Remediation Plan:

SOLVED: The Maya Protocol team added a fix to this issue in the HAL-02 branch. The calling code will refuse to add liquidity when the Pool status is not Available. This will prevent the pool from being modified.

The commit, including this change, can be viewed by clicking here.

3.2 (HAL-02) LIQUIDITY PROVIDERS MAY LOSE FUNDS WHEN DONATING TO UNAVAILABLE POOLS - HIGH

Description:

The addLiquidityFromDonate function processes liquidity donations. Most of the logic for processing the liquidity addition relies on the pool.Status field being equal to a PoolAvailable enum value.

However, the code will update a Liquidity Provider's information even when the Pool is not available (i.e. when it has a status other than PoolAvailable. This is done by calling SetLiquidityProvider on Line 208.

This may have unintended negative consequences for the Maya Protocol, including loss of fund. It may also lead to unusual program states, such as the creation of liquidity provider units for suspended pools.

During testing, it was confirmed that the Liquidity Providers' "pending assets" disappear when a donation is made to an unavailable pool without a corresponding increase in their "deposit assets".

Code Location:

x/thorchain/handler_donate.go, Lines 116-208

Listing 2: The LiquidityProvider su is modified even when the Pool is not available. (Lines 116,126,127,128,160,161,208)

```
116 func (h DonateHandler) addLiquidityFromDonate(ctx cosmos.Context,
 ↳ msg MsgDonate, pool types.Pool, su types.LiquidityProvider) error
 ↳ {
117     ver := h.mgr.Keeper().GetLowestActiveVersion(ctx)
118     synthSupply := h.mgr.Keeper().GetTotalSupply(ctx, pool.Asset.
 ↳ GetSyntheticAsset())
119     originalUnits := pool.CalcUnits(ver, synthSupply)
120
```

```
121     pendingAssetAmt := su.PendingAsset
122     pendingRuneAmt := common.GetSafeShare(pendingAssetAmt, pool.
 ↳ PendingInboundAsset, msg.RuneAmount)
123
124     pool.PendingInboundRune = common.SafeSub(pool.
 ↳ PendingInboundRune, su.PendingRune)
125     pool.PendingInboundAsset = common.SafeSub(pool.
 ↳ PendingInboundAsset, su.PendingAsset)
126     su.PendingAsset = cosmos.ZeroUint()
127     su.PendingRune = cosmos.ZeroUint()
128     su.PendingTxID = ""
129
130     ctx.Logger().Info("pre add liquidity", "pool", pool.Asset, "
 ↳ rune", pool.BalanceRune, "asset", pool.BalanceAsset, "LP units",
 ↳ pool.LPUnits, "synth units", pool.SynthUnits)
131
132     balanceRune := pool.BalanceRune
133     balanceAsset := pool.BalanceAsset
134
135     oldPoolUnits := pool.GetPoolUnits()
136     newPoolUnits, liquidityUnits, err := calculatePoolUnitsV1(
 ↳ oldPoolUnits, balanceRune, balanceAsset, pendingRuneAmt,
 ↳ pendingAssetAmt)
137     if err != nil {
138         return ErrInternal(err, "fail to calculate pool unit")
139     }
140     ctx.Logger().Info("current pool status", "pool units",
 ↳ newPoolUnits, "liquidity units", liquidityUnits)
141     poolRune := balanceRune.Add(pendingRuneAmt)
142     poolAsset := balanceAsset.Add(pendingAssetAmt)
143     pool.LPUnits = pool.LPUnits.Add(liquidityUnits)
144     pool.BalanceRune = poolRune
145     pool.BalanceAsset = poolAsset
146     ctx.Logger().Info("post add liquidity", "pool", pool.Asset, "
 ↳ rune", pool.BalanceRune, "asset", pool.BalanceAsset, "LP units",
 ↳ pool.LPUnits, "synth units", pool.SynthUnits, "add liquidity units
 ↳ ", liquidityUnits)
147     if pool.BalanceRune.IsZero() || pool.BalanceAsset.IsZero() {
148         return ErrInternal(err, "pool cannot have zero rune or
 ↳ asset balance")
149     }
150     if err := h.mgr.Keeper().SetPool(ctx, pool); err != nil {
151         return ErrInternal(err, "fail to save pool")
152     }
```

```
153     if originalUnits.IsZero() && !pool.GetPoolUnits().IsZero() {
154         poolEvent := NewEventPool(pool.Asset, pool.Status)
155         if err := h.mgr.EventMgr().EmitEvent(ctx, poolEvent); err
 ↳ != nil {
156             ctx.Logger().Error("fail to emit pool event", "error",
 ↳  err)
157         }
158     }
159
160     su.Units = su.Units.Add(liquidityUnits)
161     if pool.Status == PoolAvailable {
162         if su.AssetDepositValue.IsZero() && su.RuneDepositValue.
 ↳ IsZero() {
163             su.RuneDepositValue = common.GetSafeShare(su.Units,
 ↳ pool.GetPoolUnits(), pool.BalanceRune)
164             su.AssetDepositValue = common.GetSafeShare(su.Units,
 ↳ pool.GetPoolUnits(), pool.BalanceAsset)
165         } else {
166             su.RuneDepositValue = su.RuneDepositValue.Add(common.
 ↳ GetSafeShare(liquidityUnits, pool.GetPoolUnits(), pool.BalanceRune
 ↳ ))
167             su.AssetDepositValue = su.AssetDepositValue.Add(common
 ↳ .GetSafeShare(liquidityUnits, pool.GetPoolUnits(), pool.
 ↳ BalanceAsset))
168         }
169         // Recalculate bond provided to node if lp is bonder
170         if LiquidityPools.Contains(pool.Asset) && !su.
 ↳ NodeBondAddress.Empty() {
171             na, err := h.mgr.Keeper().GetNodeAccount(ctx, su.
 ↳ NodeBondAddress)
172             if err != nil {
173                 ctx.Logger().Error("fail to get bonded node
 ↳ account of LP %s", su.RuneAddress)
174             }
175
176             bp, err := h.mgr.Keeper().GetBondProviders(ctx, su.
 ↳ NodeBondAddress)
177             if err != nil {
178                 ctx.Logger().Error("fail to get bonded providers
 ↳ for node account of LP %s", su.RuneAddress)
179             }
180
181             addedUnits := su.Units.Sub(originalUnits)
```

```
182             addedBond := common.GetSafeShare(addedUnits, pool.
 ↳ LPUnits, pool.BalanceRune)
183             bondEvent := NewEventBond(addedBond, BondPaid, msg.Tx)
184             na.Bond = na.Bond.Add(addedBond)
185
186             from, err := su.RuneAddress.AccAddress()
187             if err != nil {
188                 ctx.Logger().Error("fail to get lp account", "
 ↳ error", err)
189             }
190
191             if bp.Has(from) {
192                 bp.BondLiquidity(from)
193             }
194
195             if err := h.mgr.EventMgr().EmitEvent(ctx, bondEvent);
 ↳ err != nil {
196                 ctx.Logger().Error("fail to emit bond event", "
 ↳ error", err)
197             }
198
199             if err := h.mgr.Keeper().SetNodeAccount(ctx, na); err
 ↳ != nil {
200                 return ErrInternal(err, fmt.Sprintf("fail to save
 ↳ node account(%s)", na.String()))
201             }
202
203             if err := h.mgr.Keeper().SetBondProviders(ctx, bp);
 ↳ err != nil {
204                 return ErrInternal(err, fmt.Sprintf("fail to save
 ↳ bond providers(%s)", bp.NodeAddress.String()))
205             }
206         }
207     }
208     h.mgr.Keeper().SetLiquidityProvider(ctx, su)
```

This was tested by modifying existing unit tests in the file x/thorchain/handle_donate_test.go by changing the p.Status value to PoolSuspended. The result indicates that the value of PendingAsset is lost, and LPUnits are added. The Pool itself is not modified.

Unit test results for the TestDonateLiquidityAuction test in the x/

thorchain/handle_donate_test.go file. Line 148 is modified to p.Status = PoolSuspended. Printed statements are included below for clarity:

Listing 3: PendingAsset removed and LPUnits added for LP interacting with a suspended Pool. Asset donations appear to be lost as they are not tracked as deposits

```
== BEFORE donate ==
LP info- PendingRune: 0 PendingAsset: 1000000000 RuneDepositValue:
↳  0 AssetDepositValue: 0 LPUnits: 0

LP info- PendingRune: 0 PendingAsset: 1000000000 RuneDepositValue:
↳  0 AssetDepositValue: 0 LPUnits: 0

LP info- PendingRune: 0 PendingAsset: 1000000000 RuneDepositValue:
↳  0 AssetDepositValue: 0 LPUnits: 0

LP info- PendingRune: 0 PendingAsset: 1000000000 RuneDepositValue:
↳  0 AssetDepositValue: 0 LPUnits: 0

== AFTER donate ==
LP info- PendingRune: 0 PendingAsset: 0 RuneDepositValue: 0
↳ AssetDepositValue: 0 LPUnits: 503246753

----------------------------------------------------------------------
↳
FAIL: <autogenerated>:1: HandlerDonateSuite.
↳ TestDonateLiquidityAuction

handler_donate_test.go:225:
    c.Assert(lp.RuneDepositValue.IsZero(), Equals, false)
... obtained bool = true
... expected bool = false
```

Recommendation:

Evaluate whether this behavior is intentional and what impacts it may have on the Liquidity Auction system. Consider either:

1. Incrementing the Deposit values for suspended Pools.
2. Disabling donations to pools that are unavailable.

Remediation Plan:

SOLVED: The Maya Protocol team added a fix to this issue in the HAL-02 branch. The calling code will refuse to add liquidity when the Pool status is not Available. This will prevent any loss of assets.

The merge request including this change can be viewed by clicking here.

3.3 (HAL-03) USE OF CRITICALLY VULNERABLE COSMOSSDK VERSION - MEDIUM

Description:

The CosmosSDK version used is affected by a critical security vulnerability that impacts all IBC-enabled Cosmos chains, for all versions of IBC. The vulnerability was outlined in the 'Dragonberry' security advisory posted by the Cosmos maintainers.

There is no public exploit for this vulnerability at this time. However, it is related to the high-profile hack of the BSC ecosystem and as a result, this code is likely to be a target for attackers in the future.

Code Location:

go.mod, Line 14

Listing 4: CosmosSDK version in use

```
14     github.com/cosmos/cosmos-sdk v0.45.1
```

Recommendation:

Update the CosmosSDK version to at least 0.45.9 in order to remediate this issue. Please read the release notes as some additional configuration must be performed in order to fix the issue. Specifically, the following line must be changed in the go.mod file:

Listing 5: The following line must be adding to go.mod to complete the upgrade

```
1 replace github.com/confio/ics23/go => github.com/cosmos/cosmos-sdk
↳ /ics23/go v0.8.0
```

Remediation Plan:

SOLVED: The Maya Protocol team updated the CosmosSDK version in a pending merge request that can be viewed by clicking here.

3.4 (HAL-04) USE OF UNSUPPORTED GO VERSION - LOW

Description:

The CosmosSDK version used by the project is 1.17 which is out-of-date. Note that only the two most recent releases of Go are supported. For example, if the latest version is 1.19.x, Only versions 1.19.x and 1.18.x are supported.

Unsupported versions of Go will not include critical security fixes or performance enhancements.

Code Location:

Listing 6: Go version in use

```
3 go 1.17
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Consider upgrading the Go version used by the project. Note that upgrades may introduce breaking changes.

Further details can be found on the Go releases page.

Remediation Plan:

SOLVED: The Maya Protocol team updated the Go version in a pending merge request that can be viewed by clicking here.

3.5 (HAL-05) LACK OF UNIT TEST FOR LIQUIDITY POOL AFTER LIQUIDITY DONATION - LOW

Description:

The state of a Pool after a Liquidity Donation is not verified in unit tests. Manual review during the audit has shown that the Pools appear to be updated correctly. However, bugs may be introduced in further development that could cause the Pool's values to become inaccurate. The lack of unit tests on the state of the Pool could lead to such bugs going undetected in a production environment until there is some loss of user funds.

Code Location:

x/thorchain/handler_donate_test.go contains unit tests for the Liquidity Auction feature. It does not contain code that verifies the status of the Pool after donations.

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Add a unit test to check the status of the Pool. Check both the case when a Liquidity Donation is successful and when it fails. The presence of a unit test will help detect bugs during the development phase rather than in a live context.

Remediation Plan:

SOLVED: The Maya Protocol team added a merge request that contains a unit test for this case. The unit test can be viewed by clicking here.

3.6 (HAL-06) MNEMONIC PHRASES ARE EXPOSED THROUGH THE REPOSITORY - LOW

Description:

The use of a hard-coded cryptographic key significantly increases the possibility that encrypted data may be recovered.

Code Location:

bifrost/thorclient/helpers_test.go

Listing 7

```
35 "industry segment educate height inject hover bargain offer employ
↳  select speak outer video tornado story slow chief object junk
↳ vapor venue large shove behave"
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Use a secure vault to store credentials instead of including them directly in the code. Also, it is recommended to delete mnemonic phrases from the Git history, and change any previously used mnemonic phrases to prevent future incidents if these mnemonic phrases have been leaked, or reused elsewhere. Test scripts should dynamically create keys at runtime.

Remediation Plan:

RISK ACCEPTED: The Maya Protocol team accepted the risk of this finding.

3.7 (HAL-07) UNCHECKED ERRORS - LOW

Description:

Unchecked error conditions were identified in numerous locations in the codebase. Unchecked errors can lead to undefined program states and interrupt normal use of the software.

Code Location:

In the example below, the removeBondAddress function is called. Although this function can return an error, there is no check for this case.

Listing 8

```
1 x/thorchain/handler_unbond.go:235:22:   removeBondAddress(ctx, h.
↳ mgr, common.Address(provider.BondAddress.String()))
2 x/thorchain/handler_unbond.go:243:21:   removeBondAddress(ctx, h.
↳ mgr, common.Address(msg.BondProviderAddress.String()))
3 x/thorchain/handler_unbond.go:267:20:   removeBondAddress(ctx, h.
↳ mgr, common.Address(from.String()))
```

It is important to note that there are several other instances of unchecked errors in the codebase. See the AUTOMATED TESTING section at the end of this report for a sample output of the errcheck tool that reports instances of unchecked errors.

Recommendation:

Handle errors wherever they can occur in order to safely handle all error states of the program. In addition, consider integrating errcheck or a similar tool in the testing pipeline so that potential errors can be flagged and managed before code is released in a production environment.

Remediation Plan:

SOLVED: The Maya Protocol team has added error handling for the cases listed above, as well as in other locations. The merge request containing these changes can be viewed here.

3.8 (HAL-08) MISLEADING FUNCTION NAME - INFORMATIONAL

Description:

The function canMsgInLiquidityAuction appears to check whether a given message can be used during a Liquidity Auction. When the function returns true, this function returns an error saying the message cannot be processed.

This could lead to developer confusion as the call to the canMsgInLiquidityAuction, when it returns true, results in an error that states that the message cannot be processed.

Code Location:

x/thorchain/handler_deposit Lines 193-195

Listing 9

```
193     if canMsgInLiquidityAuction(ctx, m, h.mgr) {
194         return nil, errors.New(fmt.Sprintf("liquidity auction
 ↳ enabled, can't process msg"))
195     }
```

Recommendation:

Refactor the code so that the names indicate the expected behavior. Consider renaming the function, re-wording the error message, and/or adding a code comment explaining the behavior.

Remediation Plan:

SOLVED: The function name has been changed in the following merged request.

3.9 (HAL-09) SPELLING MISTAKES - INFORMATIONAL

Description:

Spelling mistakes were identified within the codebase.

Code Location:

x/thorchain/helpers.go, Line 749

Listing 10

```
749 ctx.Logger().Error("failt to parse address for trading halt check"
 ↳ , "address", raw, "error", err)
```

README.md, Line 89

Listing 11

```
 89 provier units awarded to a liquidity provider is given by:
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

- Instances of failt should be changed to failed.
- Instances of provier should be changed to provider.

It is recommended that all filenames and code comments are spelled correctly, as this will help to prevent confusion during development.

Proper spelling can help convey a sense of professionalism to various project stakeholders.

Remediation Plan:

SOLVED: The spelling mistakes have been corrected in a merge request.

AUTOMATED TESTING

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**, **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.

CodeQL - Security Analysis Output Sample:

Severity : error [0]

Severity : warning [7]

- crypto-com/cosmos-sdk-codeql/beginendblock-panic Possible panics in BeginBock- or EndBlock-related consensus methods could cause a chain halt: **443**
- crypto-com/cosmos-sdk-codeql/goroutine Spawning a Go routine may be a possible source of non-determinism: **33**
- crypto-com/cosmos-sdk-codeql/map-iteration Iteration over map may be a possible source of non-determinism: **26**
- crypto-com/cosmos-sdk-codeql/floating-point-arithmetic Floating point arithmetic operations are not associative and a possible source of non-determinism: **25**
- crypto-com/cosmos-sdk-codeql/system-time Calling the system time may be a possible source of non-determinism: **23**
- crypto-com/cosmos-sdk-codeql/bech-32-constant Directly using the bech32 constants instead of the configuration values: **14**
- crypto-com/cosmos-sdk-codeql/sensitive-import Certain system packages contain functions which may be a possible source of non-determinism: **3**

Severity : note [0]

Figure 1: CodeQL output sample for the Maya Protocol modules

Gosec - Security Analysis Output Sample:

```
[/Users/user/Documents/halborn/projects/maya-liquidity-auction/src/x/thorchain/types/type_tss.go:93-97]
- G705 (CWE-): the value in the range statement should be _ unless copying a map: want: for key := range
 m (Confidence: MEDIUM, Severity: HIGH)
    92:     // analyze-ignore(map-iteration)
  > 93:     for chain, count := range chainCount {
  > 94:         if HasSuperMajority(count, len(m.PubKeys)) {
  > 95:             chains = append(chains, chain)
  > 96:         }
  > 97:     }
    98:


[/Users/user/Documents/halborn/projects/maya-liquidity-auction/src/x/thorchain/types/type_mimir.go:79-83
] - G705 (CWE-): the value in the range statement should be _ unless copying a map: want: for key := ran
ge m (Confidence: MEDIUM, Severity: HIGH)
    78:     // analyze-ignore(map-iteration)
  > 79:     for val, count := range counter {
  > 80:         if maj(count, len(active)) {
  > 81:             return val, true
  > 82:         }
  > 83:     }
    84:


[/Users/user/Documents/halborn/projects/maya-liquidity-auction/src/x/thorchain/querier.go:733-738] - G70
5 (CWE-): the value in the range statement should be _ unless copying a map: want: for key := range m (C
onfidence: MEDIUM, Severity: HIGH)
    732:         // analyze-ignore(map-iteration)
  > 733:         for c, h := range chainHeights {
  > 734:             result[i].ObserveChains = append(result[i].ObserveChains, types.QueryChainHeight{
  > 735:                 Chain:  c,
  > 736:                 Height: h,
  > 737:             })
  > 738:         }
    739:
```

Figure 2: Gosec output sample for the Maya Protocol modules

Errcheck - Security Analysis Output Sample:

Listing 12

```
1 bifrost/blockscanner/blockscanner.go:275:14:    currentPos, _ := b
↳ .scannerStorage.GetScanPos() // ignore error
2 bifrost/blockscanner/blockscanner.go:289:12:    height, _ = b.
↳ thorchainBridge.GetBlockHeight()
3 bifrost/blockscanner/blockscanner.go:291:12:    height, _ = b.
↳ thorchainBridge.GetLastObservedInHeight(b.cfg.ChainID)
```

```
 4 bifrost/observer/observe.go:336:9:  name, _ := bridge.GetTHORName(
↳ raw)
 5 bifrost/observer/observe.go:528:14: chainAddr, _ := item.
↳ ObservedVaultPubKey.GetAddress(txIn.Chain)
 6 bifrost/pkg/chainclients/binance/binance.go:292:8:  acct, _ := b.
↳ GetAccountByAddress(addr, nil)
 7 bifrost/pkg/chainclients/binance/binance.go:524:23: defer resp.
↳ Body.Close()
 8 bifrost/pkg/chainclients/binance/binance_block_scanner.go:145:23:
↳   defer resp.Body.Close()
 9 bifrost/pkg/chainclients/binance/binance_block_scanner.go:289:2:
↳    _ = json.Unmarshal(buf, &errorBlock) // ignore error
10 bifrost/pkg/chainclients/binance/binance_block_scanner.go:324:5:
↳    u, _ := url.Parse(b.cfg.RPCHost)
11 bifrost/pkg/chainclients/signercache/signer_cache_store.go:49:9:
↳    exist, _ := s.db.Has([]byte(key), nil)
12 bifrost/pkg/chainclients/terra/cosmos_client.go:197:28: c.
↳ cosmosScanner.grpc.Close()
13 bifrost/pkg/chainclients/terra/util.go:123:11:  host, _, _ = net.
↳ SplitHostPort(url.Host)
14 bifrost/pkg/chainclients/terra/util.go:135:22:  defer jsonFile.
↳ Close()
15 bifrost/pubkeymanager/mock_pool_address_validator.go:27:11: pubKey
↳ , _ := common.NewPubKey(validpb)
16 bifrost/pubkeymanager/mock_pool_address_validator.go:40:10: pubKey
↳ , _ := common.NewPubKey(validpb)
17 bifrost/signer/storage.go:53:7: buf, _ := json.Marshal(struct {
18 bifrost/signer/storage.go:180:6:    ok, _ = s.db.Has([]byte(key),
↳ nil)
19 bifrost/thorclient/broadcast.go:87:15:  _, seqNum, _ := b.
↳ getAccountNumberAndSequenceNumber()
20 bifrost/thorclient/types/rpc_block.go:25:5: u, _ := url.Parse(
↳ rpcHost)
21 cmd/thornode/cmd/pubkey.go:58:10:   prefix, _ := cmd.Flags().
↳ GetString(keys.FlagBechPrefix)
22 cmd/thornode/cmd/pubkey.go:65:15:   fmt.Fprintln(cmd.OutOrStdout()
↳ , pubKey)
23 cmd/thornode/cmd/pubkey.go:71:15:   fmt.Fprintln(cmd.OutOrStdout()
↳ , pubKey)
24 common/address.go:225:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
25 common/address.go:229:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
```

```
26 common/address.go:232:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
27 common/address.go:326:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
28 common/address.go:336:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
29 common/address.go:347:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
30 common/address.go:370:14:   prefix, _, _ := bech32.Decode(addr.
↳ String())
31 common/cosmos/cosmos.go:147:13: username, _ = input.GetString("
↳ Enter Signer name:", reader)
32 common/cosmos/cosmos.go:153:13: password, _ = input.GetPassword("
↳ Enter Signer password:", reader)
33 common/encryption.go:26:9:  block, _ := aes.NewCipher([]byte(hash)
↳ )
34 common/node-relay.go:95:12: postBody, _ := json.Marshal(n)
35 common/node-relay.go:105:23:    defer resp.Body.Close()
36 common/symbol.go:35:10: ticker, _ := NewTicker(parts[0])
37 tools/extract/extract.go:38:11: privKey, _ := keyManager.
↳ ExportAsPrivateKey()
38 tools/generate/generate.go:38:8:    conv, _ := bech32.ConvertBits(
↳ keyBytes, 8, 5, true)
39 tools/generate/generate.go:39:10:   pubKey, _ := bech32.Encode("
↳ bnbp", conv)
40 x/thorchain/client/rest/broadcast.go:108:4: _ = builder.
↳ SetSignatures()
41 x/thorchain/client/rest/query.go:18:14: fmt.Fprintf(w, `{"ping":"
↳ pong"}`)
42 x/thorchain/client/rest/query.go:49:6:  _, _ = w.Write(res)
43 x/thorchain/genesis.go:253:10:  mimir, _ := common.NewAsset("MAYA.
↳ MIMIR")
44 x/thorchain/genesis.go:271:15:  reserveAddr, _ := keeper.
↳ GetModuleAddress(ReserveName)
45 x/thorchain/genesis.go:273:12:  bondAddr, _ := keeper.
↳ GetModuleAddress(BondName)
46 x/thorchain/genesis.go:275:14:  asgardAddr, _ := keeper.
↳ GetModuleAddress(AsgardName)
47 x/thorchain/genesis.go:277:12:  mayaAddr, _ := keeper.
↳ GetModuleAddress(MayaFund)
48 x/thorchain/genesis.go:286:22:  defer iterator.Close()
49 x/thorchain/genesis.go:301:22:  defer iterator.Close()
50 x/thorchain/genesis.go:327:22:  defer iterator.Close()
51 x/thorchain/genesis.go:417:23:  defer iterVault.Close()
```

```
52 x/thorchain/genesis.go:429:25:  defer iterMsgSwap.Close()
53 x/thorchain/genesis.go:438:28:  defer iterNetworkFee.Close()
54 x/thorchain/genesis.go:447:18:  defer iter.Close()
55 x/thorchain/genesis.go:456:23:  defer iterNames.Close()
56 x/thorchain/genesis.go:464:23:  defer mimirIter.Close()
57 x/thorchain/handler.go:322:22:  defer iterator.Close()
58 x/thorchain/handler_archive.go:20:22:   defer iterator.Close()
59 x/thorchain/handler_ban.go:125:37:  h.mgr.Slasher().
↳ SlashNodeAccountLP(ctx, banner, slashAmount)
60 x/thorchain/handler_deposit.go:98:8:    memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), msg.Memo) // ignore
↳ err
61 x/thorchain/handler_deposit_archive.go:62:8:    memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), msg.Memo) // ignore
↳ err
62 x/thorchain/handler_donate.go:84:23:    defer iterator.Close()
63 x/thorchain/handler_errata_tx.go:174:8: memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), tx.Memo)
64 x/thorchain/handler_manage_thorname.go:114:10:  enable, _ := h.mgr
↳ .Keeper().GetMimir(ctx, "THORNames")
65 x/thorchain/handler_migrate.go:87:16:   fromAddress, _ := tx.
↳ VaultPubKey.GetAddress(tx.Chain)
66 x/thorchain/handler_mimir.go:93:4:  _ = h.mgr.Keeper().DeleteMimir
↳ (ctx, msg.Key)
67 x/thorchain/handler_observed_txin.go:193:9: memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), tx.Tx.Memo) // ignore
↳ err
68 x/thorchain/handler_observed_txin_archive.go:72:9:  memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), tx.Tx.Memo) // ignore
↳ err
69 x/thorchain/handler_observed_txout.go:175:9:    memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), tx.Tx.Memo)
70 x/thorchain/handler_observed_txout_archive.go:58:9: memo, _ :=
↳ ParseMemoWithTHORNames(ctx, h.mgr.Keeper(), tx.Tx.Memo)
71 x/thorchain/handler_ragnarok.go:97:17:  fromAddress, _ := tx.
↳ VaultPubKey.GetAddress(tx.Chain)
72 x/thorchain/handler_unbond.go:235:22:   removeBondAddress(ctx, h.
↳ mgr, common.Address(provider.BondAddress.String()))
73 x/thorchain/handler_unbond.go:243:21:   removeBondAddress(ctx, h.
↳ mgr, common.Address(msg.BondProviderAddress.String()))
74 x/thorchain/handler_unbond.go:267:20:   removeBondAddress(ctx, h.
↳ mgr, common.Address(from.String()))
75 x/thorchain/handler_yggdrasil.go:146:16:    fromAddress, _ := tx.
↳ VaultPubKey.GetAddress(tx.Chain)
```

```
76 x/thorchain/helpers.go:325:34:  mgr.Slasher().SlashNodeAccountLP(
↳ ctx, *nodeAcc, slashRune)
77 x/thorchain/helpers.go:499:22:  defer iterator.Close()
78 x/thorchain/helpers.go:612:23:  defer vaultIter.Close()
79 x/thorchain/helpers.go:635:22:  defer iterator.Close()
80 x/thorchain/helpers.go:889:8:   busd, _ := common.NewAsset("BNB.
↳ BUSD-BD1")
81 x/thorchain/helpers.go:890:8:   usdc, _ := common.NewAsset("ETH.
↳ USDC-0XA0B86991C6218B36C1D19D4A2E9EB0CE3606EB48")
82 x/thorchain/helpers.go:891:8:   usdt, _ := common.NewAsset("ETH.
↳ USDT-0XDAC17F958D2EE523A2206206994597C13D831EC7")
83 x/thorchain/helpers.go:1057:11: asgards, _ := mgr.Keeper().
↳ GetAsgardVaults(ctx)
84 x/thorchain/helpers.go:1104:22: defer iterator.Close()
85 x/thorchain/helpers.go:1119:11: memo, _ := ParseMemo(mgr.
↳ GetVersion(), tx.Memo)
86 x/thorchain/keeper/v1/keeper_last_height.go:14:14:  lastHeight, _
↳ := k.GetLastSignedHeight(ctx)
87 x/thorchain/keeper/v1/keeper_last_height.go:34:14:  lastHeight, _
↳ := k.GetLastChainHeight(ctx, chain)
88 x/thorchain/keeper/v1/keeper_last_height.go:54:18:  defer iter.
↳ Close()
89 x/thorchain/keeper/v1/keeper_last_height.go:91:18:  defer iter.
↳ Close()
90 x/thorchain/keeper/v1/keeper_mimir.go:41:5: _, _ = k.getInt64(ctx,
↳  k.GetKey(ctx, prefixMimir, KRAKEN), &record)
91 x/thorchain/keeper/v1/keeper_mimir.go:107:5:    _, _ = k.getInt64(
↳ ctx, k.GetKey(ctx, prefixNodePauseChain, acc.String()), &record)
92 x/thorchain/keeper/v1/keeper_node_account.go:52:24: defer
↳ naIterator.Close()
93 x/thorchain/keeper/v1/keeper_node_account.go:73:24: defer
↳ naIterator.Close()
94 x/thorchain/keeper/v1/keeper_node_account.go:104:3: _ = dbError(
↳ ctx, "Unable to list active node accounts", err)
95 x/thorchain/keeper/v1/keeper_node_account.go:157:3: _ = dbError(
↳ ctx, "Unable to list active node accounts", err)
96 x/thorchain/keeper/v1/keeper_node_account.go:215:18:    defer iter
↳ .Close()
97 x/thorchain/keeper/v1/keeper_pool.go:43:22: defer iterator.Close()
98 x/thorchain/keeper/v1/keeper_thorname.go:50:11: record, _ = k.
↳ GetTHORName(ctx, name)
99 x/thorchain/keeper/v1/keeper_txin.go:78:5:  _, _ = k.getStrings(
↳ ctx, key, &record)
```

```
100 x/thorchain/keeper/v1/keeper_txin.go:94:5:  _, _ = k.getStrings(
 ↳ ctx, key, &strs)
101 x/thorchain/keeper/v1/keeper_txout.go:83:5: _ = dbError(ctx, fmt.
 ↳ Sprintf("unable to get pool : %s", item.Coin.Asset), err)
102 x/thorchain/keeper/v1/keeper_vault.go:215:22:   defer iterator.
 ↳ Close()
103 x/thorchain/manager_network_current.go:553:22:  defer iterator.
 ↳ Close()
104 x/thorchain/manager_network_current.go:706:22:  defer iterator.
 ↳ Close()
105 x/thorchain/manager_network_v76.go:558:22:  defer iterator.Close()
106 x/thorchain/manager_network_v76.go:751:22:  defer iterator.Close()
107 x/thorchain/manager_network_v87.go:561:22:  defer iterator.Close()
108 x/thorchain/manager_network_v87.go:723:22:  defer iterator.Close()
109 x/thorchain/manager_observer_current.go:70:10:  addr, _ := cosmos.
 ↳ AccAddressFromBech32(key)
110 x/thorchain/manager_slasher_current.go:245:10:  memo, _ :=
 ↳ ParseMemoWithTHORNames(ctx, s.keeper, tx.Memo) // ignore err
111 x/thorchain/manager_swap_queue_current.go:103:22:   defer iterator
 ↳ .Close()
112 x/thorchain/manager_txout_current.go:54:27: maxGasCache[tx.Chain],
 ↳  _ = mgr.GasMgr().GetMaxGas(ctx, tx.Chain)
113 x/thorchain/manager_txout_current.go:96:9:  items, _ := tos.
 ↳ GetOutboundItems(ctx)
114 x/thorchain/manager_txout_current.go:107:2: _ = tos.keeper.
 ↳ ClearTxOut(ctx, common.BlockHeight(ctx))
115 x/thorchain/manager_txout_current.go:533:8: memo, _ := ParseMemo(
 ↳ mgr.GetVersion(), item.Memo) // ignore err
116 x/thorchain/manager_txout_current.go:551:8: memo, _ := ParseMemo(
 ↳ version, toi.Memo) // ignore err
117 x/thorchain/manager_txout_v78.go:53:27: maxGasCache[tx.Chain], _ =
 ↳  mgr.GasMgr().GetMaxGas(ctx, tx.Chain)
118 x/thorchain/manager_txout_v78.go:90:9:  items, _ := tos.
 ↳ GetOutboundItems(ctx)
119 x/thorchain/manager_txout_v78.go:101:2: _ = tos.keeper.ClearTxOut(
 ↳ ctx, common.BlockHeight(ctx))
120 x/thorchain/manager_txout_v78.go:517:8: memo, _ := ParseMemo(mgr.
 ↳ GetVersion(), item.Memo) // ignore err
121 x/thorchain/manager_txout_v78.go:531:8: memo, _ := ParseMemo(tos.
 ↳ keeper.Version(), toi.Memo) // ignore err
122 x/thorchain/manager_txout_v83.go:53:27: maxGasCache[tx.Chain], _ =
 ↳  mgr.GasMgr().GetMaxGas(ctx, tx.Chain)
123 x/thorchain/manager_txout_v83.go:90:9:  items, _ := tos.
 ↳ GetOutboundItems(ctx)
```

```
124 x/thorchain/manager_txout_v83.go:101:2: _ = tos.keeper.ClearTxOut(
 ↳ ctx, common.BlockHeight(ctx))
125 x/thorchain/manager_txout_v83.go:520:8: memo, _ := ParseMemo(mgr.
 ↳ GetVersion(), item.Memo) // ignore err
126 x/thorchain/manager_txout_v83.go:534:8: memo, _ := ParseMemo(
 ↳ version, toi.Memo) // ignore err
127 x/thorchain/manager_txout_v84.go:53:27: maxGasCache[tx.Chain], _ =
 ↳  mgr.GasMgr().GetMaxGas(ctx, tx.Chain)
128 x/thorchain/manager_txout_v84.go:90:9:  items, _ := tos.
 ↳ GetOutboundItems(ctx)
129 x/thorchain/manager_txout_v84.go:101:2: _ = tos.keeper.ClearTxOut(
 ↳ ctx, common.BlockHeight(ctx))
130 x/thorchain/manager_txout_v84.go:512:8: memo, _ := ParseMemo(mgr.
 ↳ GetVersion(), item.Memo) // ignore err
131 x/thorchain/manager_txout_v84.go:526:8: memo, _ := ParseMemo(
 ↳ version, toi.Memo) // ignore err
132 x/thorchain/manager_txout_v85.go:53:27: maxGasCache[tx.Chain], _ =
 ↳  mgr.GasMgr().GetMaxGas(ctx, tx.Chain)
133 x/thorchain/manager_txout_v85.go:90:9:  items, _ := tos.
 ↳ GetOutboundItems(ctx)
134 x/thorchain/manager_txout_v85.go:101:2: _ = tos.keeper.ClearTxOut(
 ↳ ctx, common.BlockHeight(ctx))
135 x/thorchain/manager_txout_v85.go:522:8: memo, _ := ParseMemo(mgr.
 ↳ GetVersion(), item.Memo) // ignore err
136 x/thorchain/manager_txout_v85.go:536:8: memo, _ := ParseMemo(
 ↳ version, toi.Memo) // ignore err
137 x/thorchain/manager_validator_current.go:752:35:    mgr.Slasher().
 ↳ SlashNodeAccountLP(ctx, na, amt)
138 x/thorchain/manager_validator_current.go:977:18:    defer iter.
 ↳ Close()
139 x/thorchain/manager_validator_current.go:1015:18:   defer iter.
 ↳ Close()
140 x/thorchain/manager_validator_current.go:1317:11:   status, _ :=
 ↳ vm.NodeAccountPreflightCheck(ctx, na, constAccessor)
141 x/thorchain/manager_validator_v84.go:759:35:    mgr.Slasher().
 ↳ SlashNodeAccountLP(ctx, na, amt)
142 x/thorchain/manager_validator_v84.go:988:18:    defer iter.Close()
143 x/thorchain/manager_validator_v84.go:1026:18:   defer iter.Close()
144 x/thorchain/manager_validator_v84.go:1315:11:   status, _ := vm.
 ↳ NodeAccountPreflightCheck(ctx, na, constAccessor)
145 x/thorchain/manager_yggdrasil_current.go:236:11:    addr, _ := ygg
 ↳ .PubKey.GetThorAddress()
146 x/thorchain/manager_yggdrasil_current.go:395:23:    defer
 ↳ vaultIter.Close()
```

```
147 x/thorchain/querier.go:255:18:  defer iter.Close()
148 x/thorchain/querier.go:332:18:  defer iter.Close()
149 x/thorchain/querier.go:773:22:  defer iterator.Close()
150 x/thorchain/querier.go:1141:22: defer iterator.Close()
151 x/thorchain/querier.go:1158:11: memo, _ := ParseMemoWithTHORNames(
 ↳ ctx, mgr.Keeper(), tx.Memo)
152 x/thorchain/querier.go:1296:18: defer iter.Close()
153 x/thorchain/querier.go:1302:22: defer iterNode.Close()
154 x/thorchain/querier.go:1328:18: defer iter.Close()
155 x/thorchain/querier.go:1349:18: defer iter.Close()
156 x/thorchain/querier.go:1377:18: defer iter.Close()
157 x/thorchain/querier.go:1407:18: defer iter.Close()
158 x/thorchain/router_upgrade_controller.go:75:23: defer vaultIter.
 ↳ Close()
159 x/thorchain/swap_current.go:78:15:  burnHeight, _ := keeper.
 ↳ GetMimir(ctx, "BurnSynths")
160 x/thorchain/swap_current.go:84:15:  mintHeight, _ := keeper.
 ↳ GetMimir(ctx, "MintSynths")
161 x/thorchain/types/test_common.go:32:5:  k, _ := cosmos.
 ↳ Bech32ifyPubKey(cosmos.Bech32PubKeyTypeConsPub, accts[0].PubKey)
162 x/thorchain/types/test_common.go:37:8:  addr, _ := pubKeys.
 ↳ Secp256k1.GetThorAddress()
163 x/thorchain/types/test_common.go:57:5:  k, _ := cosmos.
 ↳ Bech32ifyPubKey(cosmos.Bech32PubKeyTypeConsPub, accts[0].PubKey)
164 x/thorchain/types/test_common.go:62:8:  addr, _ := pubKeys.
 ↳ Secp256k1.GetThorAddress()
165 x/thorchain/types/test_common.go:141:7: str, _ := common.
 ↳ ConvertAndEncode(cmd.Bech32PrefixAccAddr, crypto.AddressHash([]
 ↳ byte(name)))
166 x/thorchain/types/test_common.go:142:8: base, _ := common.
 ↳ NewAddress(str)
167 x/thorchain/types/test_common.go:149:7: str, _ := common.
 ↳ ConvertAndEncode("tbnb", crypto.AddressHash([]byte(name)))
168 x/thorchain/types/test_common.go:150:7: bnb, _ := common.
 ↳ NewAddress(str)
169 x/thorchain/types/test_common.go:157:7: str, _ := common.
 ↳ ConvertAndEncode("terra", crypto.AddressHash([]byte(name)))
170 x/thorchain/types/test_common.go:158:9: terra, _ := common.
 ↳ NewAddress(str)
171 x/thorchain/types/test_common.go:164:8: addr, _ := pubKey.
 ↳ GetAddress(common.BTCChain)
172 x/thorchain/types/test_common.go:170:8: addr, _ := pubKey.
 ↳ GetAddress(common.LTCChain)
```

```
173 x/thorchain/types/test_common.go:176:8: addr, _ := pubKey.
 ↳ GetAddress(common.DOGEChain)
174 x/thorchain/types/test_common.go:182:8: addr, _ := pubKey.
 ↳ GetAddress(common.BCHChain)
175 x/thorchain/types/test_common.go:188:10:    txHash, _ := common.
 ↳ NewTxID(common.RandStringBytesMask(64))
176 x/thorchain/types/test_common.go:204:16:    bech32PubKey, _ :=
 ↳ cosmos.Bech32ifyPubKey(cosmos.Bech32PubKeyTypeAccPub, accts[0].
 ↳ PubKey)
177 x/thorchain/types/test_common.go:205:6: pk, _ := common.NewPubKey(
 ↳ bech32PubKey)
178 x/thorchain/types/type_node_account.go:112:11:  version, _ :=
 ↳ semver.Make(m.Version)
179 x/thorchain/withdraw_current.go:63:13:  pauseAsym, _ := mgr.Keeper
 ↳ ().GetMimir(ctx, fmt.Sprintf("PauseAsymWithdrawal-%s", pool.Asset.
 ↳ Chain))
```

THANK YOU FOR CHOOSING

// HALBORN