UMEE - Oracle & Price Feeder

Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 15th, 2022 - June 3rd, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	06/01/2022	Gokberk Gulgun
0.2	Draft Review	06/07/2022	Gokberk Gulgun
0.3	Draft Review	06/07/2022	Gabi Urrutia
1.0	Remediation Plan	06/17/2022	Gokberk Gulgun
1.1	Remediation Plan Review	06/17/2022	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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

UMEE engaged Halborn to conduct a security assessment on their **Price Feeder & Oracle** implementation beginning on May 15th and ending on June 3rd, 2022.

The security assessment was scoped to the GitHub repository of UMEE. An audit of the security risk and implications regarding the changes introduced by the development team at UMEE prior to its production release, shortly following the assessment's deadline.

1.2 AUDIT SUMMARY

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

The purpose of this audit to achieve the following:

- Ensure that UMEE Oracle and Price Feeder module functions are intended.
- Identify potential security issues with the UMEE.

In summary, Halborn identified few security risks that were mostly addressed by UMEE 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 UMEE. 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 code-base.
- Ensuring correctness of the codebase.
- Dynamic Analysis on UMEE Oracle & Price Feeder module functions and data types.
- Property based coverage-guided fuzzing. (gofuzz).

1.4 SCOPE

The assessment was scoped to the repository available in GitHub at commit **d91976514c17d22198a7c1680e6d939ae7e427da**.

The audit is only scoped to the following modules :

- Price Feeder.
- Oracle.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	4	3

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - THE PRICE FEEDER IS NOT RESISTANT TO DE-PEGGING	High	SOLVED - 06/17/2022
HAL-02 - MISSING JAILED VALIDATOR CHECK	Low	RISK ACCEPTED
HAL-03 - REWARDS CAN NOT DISTRIBUTED IF THERE IS NO COINS IN THE REWARD POOL	Low	RISK ACCEPTED
HAL-04 - TWAP PERIOD IS SELECTED AS THREE MINUTE	Low	SOLVED - 06/17/2022
HAL-05 - DEPRECATED API SUPPORTED ON THE OSMOSIS	Low	SOLVED - 06/17/2022
HAL-06 - ABCI CAN BE REPLACED WITH ABCI++	Informational	ACKNOWLEDGED
HAL-07 - MISSING GOLANGCI LINT SUPPORT ON THE REPOSITORY	Informational	SOLVED - 06/17/2022
HAL-08 - PROVIDER URLS ARE HARDCODED INTO THE SYSTEM	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) THE PRICE FEEDER IS NOT RESISTANT TO DE-PEGGING - HIGH

Description:

With the recent problem of UST, the market can be volatile with the stable coins. The current price feeder does not have any protection mechanism for the de-pegging operations. If the providers are feeding with volatile(drop) price, that can cause liquidations. If the price feeder is utilizing a stable coin like UST, the votes should be converted to real USD value before submitting on chain. The many protocols are suffered from the related attack on their system. The example can be seen from the Link.

Even if external oracles like a **Chainlink** are used, during the de-pegging many protocols are affected by the wrong price feed and this caused unintended behavior on the protocols. From that reason, the price feeders should be designed with the real USD value of the assets.

Risk Level:

Likelihood - 4
Impact - 5

Recommendation:

It is recommended to feed prices through real USD value. During the recent market conditions, If the price is feed with stable coin prices, that can directly affect all modules.

Remediation Plan:

SOLVED: The UMEE Team solved this issue by implementing a requirement that any quotes for coins that are based on USD be re-denominated in USD.

Commit ID: PR FIX https://github.com/umee-network/umee/pull/925

3.2 (HAL-02) MISSING JAILED VALIDATOR CHECK - LOW

Description:

In the **MsgAggregateExchangeRatePrevote** and **MsgAggregateExchangeRateVote**, jailed validators are not checked. Jailed validators are checked in **SlashAndResetMissCounters** function. However, the check is not implemented in the messages.

Code Location:

Reference

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Even if the Validator set can be changed during the voting period, the system design should be reviewed. Jailed validator can still vote through the **MsgAggregateExchangeRatePrevote** and **MsgAggregateExchangeRateVote** messages.

Remediation Plan:

RISK ACCEPTED: The UMEE Team states that they should allow the vote, just not count it if the validator is not within the active set during counting, which they believe is what they do now. Otherwise, if the entire active set changes during a voting window, they would have no prices for the next 5 blocks.

3.3 (HAL-03) REWARDS CAN NOT DISTRIBUTED IF THERE IS NO COINS IN THE REWARD POOL - LOW

Description:

In the documentation/specs, there is no information has been found when reward pool is empty. In the reward module, there is no countermeasure mechanism has been developed for this condition.

Listing 1: Reward Distribution To Ballot Winners

```
    var periodRewards sdk.DecCoins
    for _, denom := range rewardDenoms {
        rewardPool := k.GetRewardPool(ctx, denom)
        // return if there's no rewards to give out
        if rewardPool.IsZero() {
            continue
        }

        periodRewards = periodRewards.Add(sdk.NewDecCoinFromDec(
            denom,
            sdk.NewDecFromInt(rewardPool.Amount).Mul(
↳ distributionRatio),
        ))
    }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

If the no rewards are coming from the leverage modules, the rewards cannot distribute to ballot winners. The trade-off should be considered on the related function when there are no rewards. The problem should be

mentioned on the documentation.

Remediation Plan:

RISK ACCEPTED: The UMEE Team states that they are not too concerned about the event that rewards are empty, since validators earn commission anyway, and they have a logic that would slash / jail them if they were not submitted votes.

3.4 (HAL-04) TWAP PERIOD IS SELECTED AS THREE MINUTE - LOW

Description:

TWAP or Time-weighted Average Price is a calculation that defines the weighted average price over a specified period. TWAP or Time Weighted Average Price is another variant of VWAP (Volume Weighted Average Price). During the code review, It has been observed that TWAP Candle period is selected as 3 Minute.

Risk Level:

Likelihood - 2
Impact - 2

Code Location:

Listing 2: TVWAP Candle Period

```
const (
    // tvwapCandlePeriod represents the time period we use for
↳ tvwap in minutes
    tvwapCandlePeriod = 3 * time.Minute
)

```

Recommendation:

On the several implementations, the **TWAP** minimum period has been defined as 5 minutes. The system constraints should be reviewed and the system TWAP period should be bounded by with **The minimum duration is 5 minutes, and the maximum is 24 hours**.

Remediation Plan:

SOLVED: The UMEE Team solved this issue by changing the **TWAP** period to 5 minutes.

Commit ID: PR FIX https://github.com/umee-network/umee/pull/1032/files

3.5 (HAL-05) DEPRECATED API SUPPORTED ON THE OSMOSIS - LOW

Description:

In the price feeder, the module has defined multiple provider. However, Osmosis is using deprecated APIs.

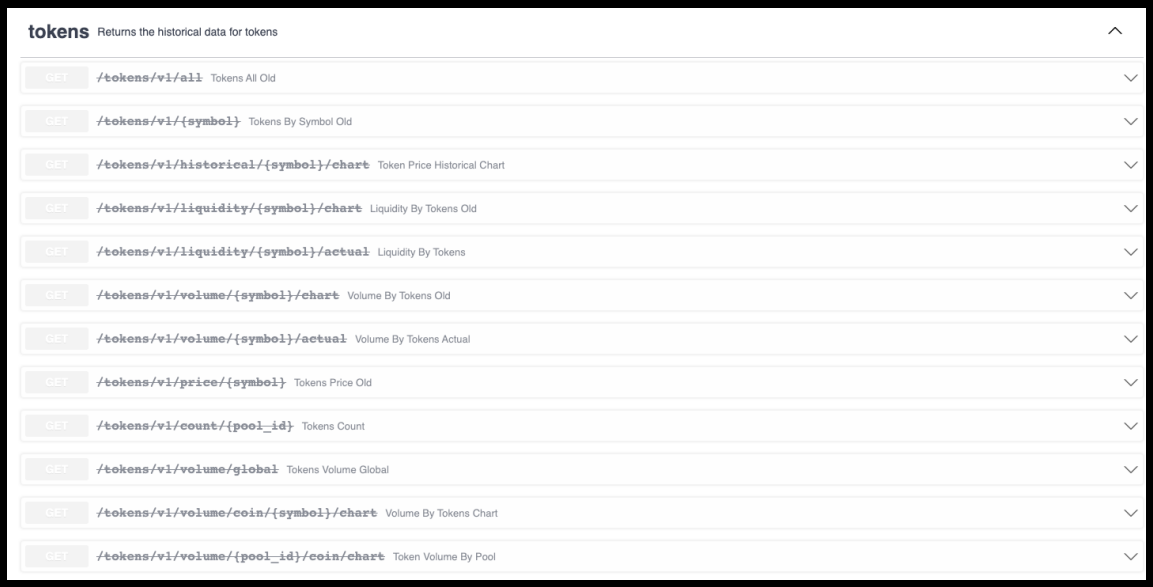
Code Location:

Listing 3: Deprecated API

```
const (
    osmosisBaseURL         = "https://api-osmosis.imperator.co"
    osmosisTokenEndpoint   = "/tokens/v1"
    osmosisCandleEndpoint  = "/tokens/v2/historical"
    osmosisPairsEndpoint   = "/pairs/v1/summary"
)
```

Risk Level:

Likelihood - 1
Impact - 3



Recommendation:

Consider to check continuously all providers on the price feeder. Deprecated provide APIs should be disabled on the protocol.

Remediation Plan:

SOLVED: The UMEE Team solved this issue by deleting deprecated APIs. The Team is working with the Osmosis Team to support web-socket APIs.

3.6 (HAL-06) ABCI CAN BE REPLACED WITH ABCI++ - INFORMATIONAL

Description:

ABCI is the interface between the consensus engine and the application. It defines when the application can talk to consensus during the execution of a blockchain. Currently, the application can only act at one phase in consensus, immediately after a block has been finalized. ABCI can only act in one phase of consensus, immediately after a block has been finalized. This restriction on the application prevents the application from implementing numerous features, including many scalability improvements, that are now better understood than when ABCI was first written. Many of the scalability proposals, for example, boil down to “Make the miners / block proposers / validators do the work, so the network doesnt have to.” Optimizations such as tx-level signature aggregation, state transition proofs, and so on are included. Furthermore, many new security properties are impossible to achieve in the current paradigm because the application cannot compel validators to do more than just finalize txs. Threshold cryptography and guaranteed IBC connection attempts are examples of such features.

ABCI++ overcomes these constraints by allowing the application to intervene at three critical points during block execution. The new interface enables block proposers to perform application-dependent work in a proposed block via the **PrepareProposal** method; validators to perform application-dependent work in a proposed block via the **ProcessProposal** method; and applications to require their validators to do more than just validate blocks via the **ExtendVote** and VerifyVoteExtension methods. Furthermore, **ABCI++** renames **BeginBlock**, **[DeliverTx]**, and **EndBlock** to **FinalizeBlock** to make it easier to deliver a decided block to the Application.

Reference:

ABCI.go

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Halborn recommends that the review the workflow on the ABCI and the take the advantage of **ABCI++**.

Remediation Plan:

ACKNOWLEDGED: The UMEE Team will implement the feature when the Cosmos SDK is fully support ABCI+.

3.7 (HAL-07) MISSING GOLANGCI LINT SUPPORT ON THE REPOSITORY - INFORMATIONAL

Description:

During the code review, It has been observed there is no GitHub action has been defined for the scanning code base with the code linters.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to use https://golangci-lint.run after the every PR/merge. The relevant findings should be fixed before the deployment.

Remediation Plan:

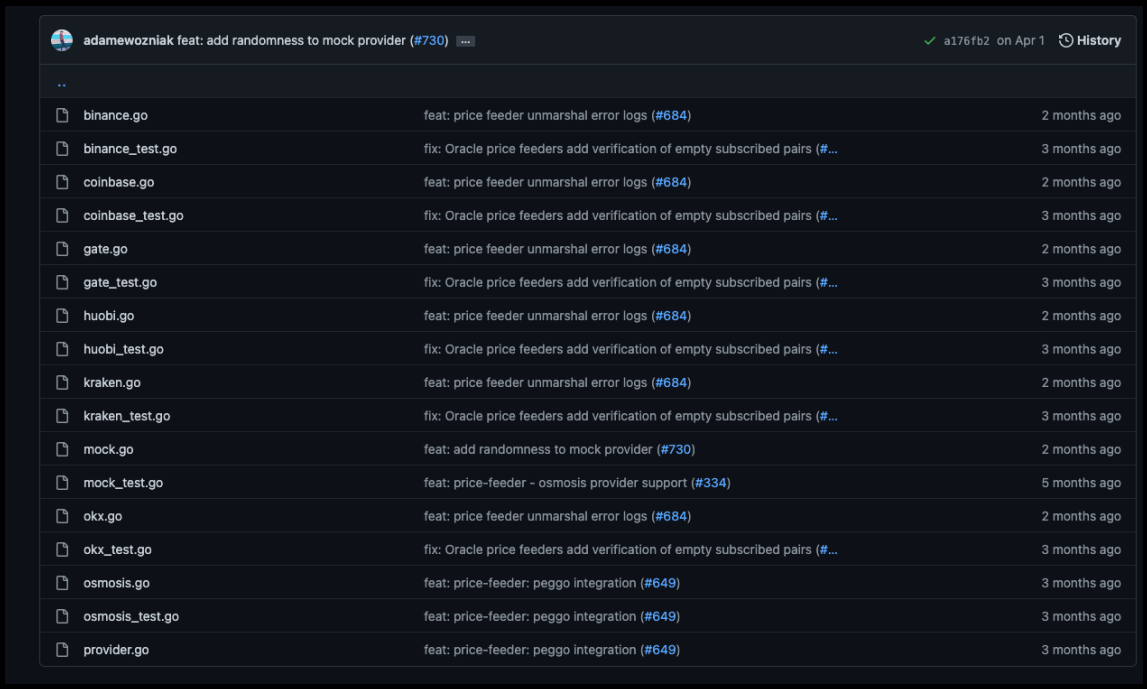
SOLVED: The UMEE Team solved this issue by adding golang-ci lint support on the repository.

3.8 (HAL-08) PROVIDER URLS ARE HARDCODED INTO THE SYSTEM - INFORMATIONAL

Description:

On the price feeders, the name separates all providers. Every provider has its configuration file. However, If the provider becomes useless, the related APIs should be updated on the system.

Location:



Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

The system design can be designed more modular. The provider APIs can be parsed from the one API and can feed to the system.

Remediation Plan:

SOLVED: The UMEE Team solved this issue by allowing to override configuration files.

Commit ID: PR FIX https://github.com/umee-network/umee/issues/1031

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 ineffassign, unconvert and LGTM. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.

Semgrep - Security Analysis Output Sample:

Listing 4: Rule Set

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

AUTOMATED TESTING

Semgrep Results:

```
% semgrep --config auto .
Fetching rules from https://semgrep.dev/registry.

Scanning across multiple languages:
    <multilang> |  62 rules × 88 files
             go | 129 rules × 37 files

  100%|

Findings:

  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

       52| protoReq.Denom, err = runtime.String(val)
         ⋮----------------------------------------
       79| protoReq.Denom, err = runtime.String(val)
         ⋮----------------------------------------
      124| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      151| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      178| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      205| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      232| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      259| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      304| protoReq.ValidatorAddr, err = runtime.String(val)
         ⋮----------------------------------------
      331| protoReq.ValidatorAddr, err = runtime.String(val)

Some files were skipped or only partially analyzed.
  Scan was limited to files tracked by git.
  Scan skipped: 17 files matching .semgrepignore patterns
  For a full list of skipped files, run semgrep with the --verbose flag.

Ran 1374 rules on 44 files: 10 findings.
```

Gosec - Security Analysis Output Sample:

```
[/x/oracle/simulations/operations.go:26] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
    25:         OpWeightMsgAggregateExchangeRateVote    = "op_weight_msg_exchange_rate_aggregate_vote"
  > 26:         OpWeightMsgDelegateFeedConsent          = "op_weight_msg_exchange_feed_consent"
    27:


[/x/oracle/simulations/operations.go:25] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
    24:         OpWeightMsgAggregateExchangeRatePrevote = "op_weight_msg_exchange_rate_aggregate_prevote"
  > 25:         OpWeightMsgAggregateExchangeRateVote    = "op_weight_msg_exchange_rate_aggregate_vote"
    26:         OpWeightMsgDelegateFeedConsent          = "op_weight_msg_exchange_feed_consent"


[/x/oracle/simulations/operations.go:24] - G101 (CWE-798): Potential hardcoded credentials (Confidence: LOW, Severity: HIGH)
    23: const (
  > 24:         OpWeightMsgAggregateExchangeRatePrevote = "op_weight_msg_exchange_rate_aggregate_prevote"
    25:         OpWeightMsgAggregateExchangeRateVote    = "op_weight_msg_exchange_rate_aggregate_vote"
```

Staticcheck - Security Analysis Output Sample:

```
x/oracle/keeper/msg_server_test.go:11:2: package "github.com/umee-network/umee/v2/x/oracle/types" is being imported more than once (ST1019)
        x/oracle/keeper/msg_server_test.go:12:2: other import of "github.com/umee-network/umee/v2/x/oracle/types"
x/oracle/keeper/msg_server_test.go:122:2: this value of err is never used (SA4006)
x/oracle/keeper/reward_test.go:29:22: this result of append is never used, except maybe in other appends (SA4010)
x/oracle/types/query.pb.gw.go:16:2: package github.com/golang/protobuf/descriptor is deprecated: See the "google.golang.org/protobuf/reflect/protoreflect" package for how to obtain an EnumDescriptor or MessageDescripto
r in order to programatically interact with the protobuf type system.  (SA1019)
x/oracle/types/query.pb.gw.go:17:2: package github.com/golang/protobuf/proto is deprecated: Use the "google.golang.org/protobuf/proto" package instead.  (SA1019)
x/oracle/types/query.pb.gw.go:33:9: descriptor.ForMessage is deprecated: Not all concrete message types satisfy the Message interface. Use MessageDescriptorProto instead. If possible, the calling code should be rewritt
en to use protobuf reflection instead. See package "google.golang.org/protobuf/reflect/protoreflect" for details.  (SA1019)
```

Unconvert - Security Analysis Output Sample:

```
/x/oracle/types/ballot_test.go:167:21: unnecessary conversion
/x/oracle/types/genesis.pb.go:426:42: unnecessary conversion
/x/oracle/types/genesis.pb.go:520:29: unnecessary conversion
/x/oracle/types/genesis.pb.go:529:26: unnecessary conversion
/x/oracle/types/oracle.pb.go:404:41: unnecessary conversion
/x/oracle/types/oracle.pb.go:433:41: unnecessary conversion
/x/oracle/types/oracle.pb.go:458:41: unnecessary conversion
/x/oracle/types/oracle.pb.go:528:41: unnecessary conversion
/x/oracle/types/oracle.pb.go:651:28: unnecessary conversion
/x/oracle/types/oracle.pb.go:658:28: unnecessary conversion
/x/oracle/types/oracle.pb.go:669:28: unnecessary conversion
/x/oracle/types/oracle.pb.go:711:28: unnecessary conversion
/x/oracle/types/oracle.pb.go:754:25: unnecessary conversion
/x/oracle/types/query.pb.go:1521:40: unnecessary conversion
/x/oracle/types/query.pb.go:1939:27: unnecessary conversion
/x/oracle/types/query.pb.go:2064:24: unnecessary conversion
/x/oracle/types/tx.pb.go:785:21: unnecessary conversion
```

THANK YOU FOR CHOOSING

// HALBORN