// HALBORN

Mars Protocol - Custom Modules Gov, Incentives & Safety

Cosmos Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 31st, 2022 - November 11th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/03/2022	John Saigle
0.2	Document Edits	11/11/2022	John Saigle
0.3	Draft Review	11/11/2022	Gabi Urrutia
1.0	Remediation Plan	11/17/2022	John Saigle
1.1	Remediation Plan Review	11/24/2022	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
John Saigle	Halborn	John.Saigle@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Mars Protocol engaged Halborn to conduct a security audit on their smart contracts beginning on October 31st, 2022 and ending on November 11th, 2022. The security assessment was scoped to the custom Cosmos modules provided to the Halborn team.

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 smart contract. 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 custom modules within the Mars Protocol repository operate as intended.
- Identify potential security issues with the implementations.

In summary, Halborn identified few security risks that were mostly addressed by Mars Protocol 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 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 gov, incentives, and safety module 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.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

This review was scoped to the Mars Protocol Hub repository.

- Repository URL: Mars Protocol repository
- Commit hash: 25ff9f1f12faeeb0c2f0173c7868b38337d095fd

Code in scope:

- x/gov
- x/incentives
- x/safety

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	3	3

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - USE OF CRITICALLY VULNERABLE COSMOS SDK VERSION	High	SOLVED - 11/16/2022
HAL-02 - PANIC IN CONSENSUS METHOD COULD LEAD TO CHAIN HALT	Low	RISK ACCEPTED
HAL-03 - USE OF VULNERABLE COMPONENTS	Low	FUTURE RELEASE
HAL-04 - ITERATION OVER MAP - SOURCE OF NON-DETERMINISM	Low	RISK ACCEPTED
HAL-05 - ERROR MESSAGE NOT DISPLAYED TO CLI USERS	Informational	SOLVED - 11/16/2022
HAL-06 - SPELLING MISTAKES	Informational	SOLVED - 11/16/2022
HAL-07 - OPEN TODOs	Informational	SOLVED - 11/16/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) USE OF CRITICALLY VULNERABLE COSMOSSDK VERSION - HIGH

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 8

Listing 1

```
8     github.com/cosmos/cosmos-sdk v0.45.6
```

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 2

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

Remediation Plan:

SOLVED: The Mars Protocol team updated to CosmosSDK version 0.45.11 and completed all necessary configurations.

3.2 (HAL-02) PANIC IN CONSENSUS FUNCTION COULD LEAD TO CHAIN HALT - LOW

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: Case 1 The BeginBlocker function in x/incentives/abci.go calls ReleaseBlockReward which can panic.

x/incentives/abci.go, Lines 17-30

Listing 3: (Line 22)

```
17 // BeginBlocker distributes block rewards to validators who have
↳ signed the previous block
18 func BeginBlocker(ctx sdk.Context, req abci.RequestBeginBlock, k
↳ keeper.Keeper) {
19     defer telemetry.ModuleMeasureSince(types.ModuleName, time.Now
↳ (), telemetry.MetricKeyBeginBlocker)
20
21     ids, totalBlockReward := k.ReleaseBlockReward(ctx, req.
↳ LastCommitInfo.Votes)
22
23     ctx.EventManager().EmitEvent(
24         sdk.NewEvent(
25             types.EventTypeIncentivesReleased,
26             sdk.NewAttribute(types.AttributeKeySchedules,
↳ marsutils.UintArrayToString(ids, ",")),
27             sdk.NewAttribute(sdk.AttributeKeyAmount,
↳ totalBlockReward.String()),
28         ),
29     )
30 }
```

ReleaseBlockReward panics if SendCoinsFromModuleToModule returns an error.

x/incentives/keeper/reward.go, Lines 22-58

Listing 4: (Lines 55,57)

```
22 func (k Keeper) ReleaseBlockReward(ctx sdk.Context, bondedVotes []
↳ abci.VoteInfo) (ids []uint64, totalBlockReward sdk.Coins) {
23     currentTime := ctx.BlockTime()
24
25     // iterate through all active schedules, sum up all rewards to
↳  be released in this block.
26     //
27     // If an incentives schedule has been fully released, delete
↳ it from the store; otherwise, update
28     // the released amount and save
29     ids = []uint64{}
30     totalBlockReward = sdk.NewCoins()
31     k.IterateSchedules(ctx, func(schedule types.Schedule) bool {
32         blockReward := schedule.GetBlockReward(currentTime)
33
34         if !blockReward.Empty() {
35             ids = append(ids, schedule.Id)
36             totalBlockReward = totalBlockReward.Add(blockReward
↳ ...)
37         }
38
39         if currentTime.After(schedule.EndTime) {
40             k.DeleteSchedule(ctx, schedule.Id)
41         } else {
42             schedule.ReleasedAmount = schedule.ReleasedAmount.Add(
↳ blockReward...)
43             k.SetSchedule(ctx, schedule)
44         }
45
46         return false
47     })
48
49     // exit here if there is no coin to be released
50     if totalBlockReward.Empty() {
51         return
```

```
52      }
53
54      // transfer the coins to distribution module account so that
↳ they can be distributed
55      err := k.bankKeeper.SendCoinsFromModuleToModule(ctx, types.
↳ ModuleName, distrtypes.ModuleName, totalBlockReward)
56      if err != nil {
57          panic(err)
58      }
```

Code Location: Case 2 The EndBlocker function in x/gov/abci.go eventually calls the function MustGetTokensInVesting in x/gov/keeper/vesting.go which interacts with a CosmWasm contract. If any errors occur when the module does this interaction, MustGetTokensInVesting will panic. This panic will cause a chain halt as it occurs in the consensus function EndBlocker

It is important to note that there is no immediate case to believe that the current CosmWasm contract will throw an error. It does not appear to be the case that an attacker could trigger an error deliberately.

The EndBlock function calls Tally which can panic
x/gov/abci.go, Lines 55. (Only the call site is shown here for brevity.)

Listing 5

```
55          passes, burnDeposits, tallyResults := keeper.Tally(ctx,
↳ proposal) // our custom implementation of tally logics
```

Tally can panic due to MustGetTokensInVesting.

x/gov/keeper/tally.go, Lines 20-45

Listing 6: (Line 45)

```
20 func (k Keeper) Tally(ctx sdk.Context, proposal govtypes.Proposal)
↳  (passes bool, burnDeposits bool, tallyResults govtypes.
↳ TallyResult) {
21      results := make(map[govtypes.VoteOption]sdk.Dec)
```

```
22      results[govtypes.OptionYes] = sdk.ZeroDec()
23      results[govtypes.OptionAbstain] = sdk.ZeroDec()
24      results[govtypes.OptionNo] = sdk.ZeroDec()
25      results[govtypes.OptionNoWithVeto] = sdk.ZeroDec()
26
27      // fetch all currently bounded validators
28      currValidators := make(map[string]govtypes.ValidatorGovInfo)
29      k.stakingKeeper.IterateBondedValidatorsByPower(ctx, func(index
↳  int64, validator stakingtypes.ValidatorI) (stop bool) {
30          currValidators[validator.GetOperator().String()] =
↳ govtypes.NewValidatorGovInfo(
31              validator.GetOperator(),
32              validator.GetBondedTokens(),
33              validator.GetDelegatorShares(),
34              sdk.ZeroDec(),
35              govtypes.WeightedVoteOptions{},
36          )
37
38          return false
39      })
40
41      // fetch all tokens locked in the vesting contract
42      //
43      // NOTE: for now we simply use the default contract address.
↳ later it may be a better idea to use
44      // a configurable parameter
45      tokensInVesting, totalTokensInVesting :=
↳ MustGetTokensInVesting(ctx, k.wasmKeeper, DefaultContractAddr)
```

MustGetTokensInVesting panics if GetTokensInVesting returns an error.

x/gov/keeper/vesting.go, Lines 89-97

Listing 7

```
89 // MustGetTokensInVesting is the same with `GetTokensInVesting`,
↳ but panics on error
90 func MustGetTokensInVesting(ctx sdk.Context, k wasmtypes.
↳ ViewKeeper, contractAddr sdk.AccAddress) (map[string]sdk.Int, sdk.
↳ Int) {
91      tokensInVesting, totalTokensInVesting, err :=
↳ GetTokensInVesting(ctx, k, contractAddr)
92      if err != nil {
```

```
93          panic(fmt.Sprintf("failed to tally vote: %s", err))
94      }
95
96      return tokensInVesting, totalTokensInVesting
97  }
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

Where possible, custom errors should be defined and handled according to the Cosmos best practices. This is preferable to using panic.

Note that in exceptional cases, panic can be used to halt execution if there is no better way to handle an error. This should be considered as a last resort and done only if there is no method to recover from the error.

Remediation Plan:

RISK ACCEPTED: The Mars Protocol team stated that a chain halt is the intended behaviour in their use case if there is an error in the consensus functions.

3.3 (HAL-03) USE OF VULNERABLE COMPONENTS - LOW

Description:

During the audit, Halborn identified installed 3rd party packages that contain known security vulnerabilities.
Packages

ID	Package	Rating	Description
sonatype-2021-0598	tendermint	MEDIUM	Improper Input Validation
CVE-2022-32149	x/text	HIGH	Uncontrolled Resource Consumption
sonatype-2022-3945	go-buffer-pool	MEDIUM	Integer Overflow

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended that all 3rd party packages that are installed are kept up to date and all security fixes applied.

Remediation Plan:

PENDING: The Mars Protocol team stated that they will make sure that all packages are updated to the latest versions.

3.4 (HAL-04) ITERATION OVER MAP - SOURCE OF NON-DETERMINISM - LOW

Description:

The codebase contains an instance of iteration over a map. As maps are not ordered in Go, iterations over maps are non-deterministic. If the consensus logic expects a deterministic result from the iteration, errors can occur where different nodes reach different states due to the differences in map ordering. If the nodes reach differing states, this can cause consensus issues, which could result in a chain halt.

Code Location:

x/gov/abci.go Line 55 calls Tally in EndBlocker

x/gov/abci.go, Lines 55. (Only the call site is shown here for brevity.)

Listing 8

```
55          passes, burnDeposits, tallyResults := keeper.Tally(ctx,
↳ proposal) // our custom implementation of tally logics
```

Tally iterates over a map currValidators.

x/gov/keeper/tally.go, Lines 99-109

Listing 9: Iteration over map (Line 100)

```
 99 // iterate over the validators again to tally their voting power
100 for _, val := range currValidators {
101     if len(val.Vote) == 0 {
102         continue
103     }
 ↳
 ↳ sharesAfterDeductions := val.DelegatorShares.Sub(val.
 ↳ DelegatorDeductions)
104     votingPower := sharesAfterDeductions.MulInt(val.BondedTokens).
```

```
 ↳ Quo(val.DelegatorShares)
105     incrementTallyResult(votingPower, val.Vote, results, &
 ↳ totalTokensVoted)
106 }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Ensure that iterations over maps do not contain any logic that depends on a fixed ordering of the map. Alternatively, make use of an alternative data structure that has a deterministic ordering or ensure that maps are sorted before iterating over them.

Remediation Plan:

RISK ACCEPTED: The Mars Protocol team has stated that they are confident that the logic does not depend on iteration order.

3.5 (HAL-05) ERROR MESSAGE NOT DISPLAYED TO CLI USERS - INFORMATIONAL

Description:

While using the CLI for the safety module, an error message is suppressed. As a result, no message is delivered to the user. This could result in a dissatisfying user experience.

Code Location:

x/safety/client/proposal_handler.go, Lines 45-47

Listing 10: (Line 46)

```
44              title, description, deposit, err := marsutils.
↳ ParseGovProposalFlags(cmd)
45              if err != nil {
46                  return nil
47              }
```

Recommendation:

Return err instead of nil.

Remediation Plan:

SOLVED: The Mars Protocol team solved this issue.

3.6 (HAL-06) SPELLING MISTAKES - INFORMATIONAL

Description:

Spelling mistakes were identified within the codebase.

Code Location:

‘Bounded’ should be ‘bonded’.
x/gov/keeper/tally.go, Line 29

Listing 11

```
29     // fetch all currently bounded validators
```

‘bondedBotes’ should be ‘bondedVotes’.
x/incentives/keeper/reward.go, Line 20

Listing 12

```
20 // `bondedBotes` is a list of {validator address, validator voted
↳ on last block flag} for all validators
```

Recommendation:

It is recommended that all filenames and in the code 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 Mars Protocol team solved this issue.

3.7 (HAL-07) 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:

Listing 13

```
x/incentives/keeper/reward.go:62  // TODO: we don't need to check
↳ `SignedLastBlock`? the allocate function in distr module doesn't
x/gov/keeper/tally.go:121:    // NOTE: should the deposit really
↳ be burned here?
```

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:

SOLVED: The Mars Protocol team solved this issue.

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 [2]

- crypto-com/cosmos-sdk-codeql/beginendblock-panic Possible panics in BeginBock- or EndBlock-related consensus methods could cause a chain halt: **19**
  - x/gov/abci.go:25
  - x/gov/abci.go:26
  - x/gov/abci.go:27
  - x/gov/abci.go:30
  - x/gov/abci.go:43
  - x/gov/abci.go:52
  - x/gov/abci.go:55
  - x/gov/abci.go:58
  - x/gov/abci.go:60
  - x/gov/abci.go:64
  - x/gov/abci.go:64
  - x/gov/abci.go:64
  - x/gov/abci.go:69
  - x/gov/abci.go:95
  - x/gov/abci.go:96
  - x/gov/abci.go:99
  - x/gov/abci.go:104
  - x/incentives/abci.go:21
  - x/incentives/abci.go:26
- crypto-com/cosmos-sdk-codeql/map-iteration Iteration over map may be a possible source of non-determinism: **2**
  - app/app.go:686
  - x/gov/keeper/tally.go:100

Figure 1: CodeQL output sample for the Mars Protocol modules

Gosec - Security Analysis Output Sample:

```
File: gosec

Results:


[/Users/user/Documents/halborn/projects/mars-hub/src/x/gov/keeper/tally.go:100-109] - G705
 (CWE-): expected exactly 1 statement (either append, delete, or copying to another map) i
n a range with a map, got 4 (Confidence: MEDIUM, Severity: HIGH)
    99:     // iterate over the validators again to tally their voting power
  > 100:     for _, val := range currValidators {
  > 101:         if len(val.Vote) == 0 {
  > 102:             continue
  > 103:         }
  > 104:
  > 105:         sharesAfterDeductions := val.DelegatorShares.Sub(val.DelegatorDeductions)
  > 106:         votingPower := sharesAfterDeductions.MulInt(val.BondedTokens).Quo(val.Dele
gatorShares)
  > 107:
  > 108:         incrementTallyResult(votingPower, val.Vote, results, &totalTokensVoted)
  > 109:     }
    110:



[/Users/user/Documents/halborn/projects/mars-hub/src/app/app.go:686-688] - G705 (CWE-): ex
pected either an append, delete, or copy to another map in a range with a map (Confidence:
 MEDIUM, Severity: HIGH)
    685:     modAccAddrs := make(map[string]bool)
  > 686:     for acc := range maccPerms {
  > 687:         modAccAddrs[authtypes.NewModuleAddress(acc).String()] = true
  > 688:     }
    689:



[/Users/user/Documents/halborn/projects/mars-hub/src/cmd/marsd/init.go:106] - G703 (CWE-):
 Returned error is not propagated up the stack. (Confidence: HIGH, Severity: LOW)
    105:             genFile := config.GenesisFile()
  > 106:             overwrite, _ := cmd.Flags().GetBool(genutilcli.FlagOverwrite)
    107:
```

Figure 2: Gosec output sample for the Mars Protocol modules

Errcheck - Security Analysis Output Sample:

Listing 14

```
cmd/marsd/init.go:79:13:        chainID, _ := cmd.Flags().
↳ GetString(flags.FlagChainID)
cmd/marsd/init.go:86:13:        recover, _ := cmd.Flags().GetBool(
↳ genutilcli.FlagRecover)
cmd/marsd/init.go:106:15:       overwrite, _ := cmd.Flags().
↳ GetBool(genutilcli.FlagOverwrite)
x/incentives/keeper/keeper.go:120:22:   defer iterator.Close()
```

THANK YOU FOR CHOOSING

// HALBORN