// HALBORN

Tenet - Node

Cosmos Security Assessment

Prepared by: **Halborn**

Date of Engagement: **September 29th, 2023 - October 30th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	10/15/2023
0.2	Document Updates	10/30/2023
0.3	Draft Review	10/30/2023
1.0	Remediation Plan	11/01/2023
1.1	Remediation Plan Review	11/01/2023

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

Tenet engaged Halborn to conduct a security assessment on their modules, beginning on September 29th, 2023 and ending on October 30th, 2023. The security assessment was scoped to the sections of code that pertain to the **Cosmos Modules/Precompiles**

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided one month for the engagement and assigned one full-time security engineer to verify the security of the merge requests. 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 assessment is to:

- Ensure that the **Tenet Modules** operate as intended.
- Identify potential security issues with the custom modules used in the SGE Chain.

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks that were mostly addressed by the Tenet 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 assessment:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (e.g., staticcheck, gosec, unconvert, codeql, ineffassign and semgrep)
- Manual Assessment for discovering security vulnerabilities on codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis on files and modules related to the **Subaccount Module**.

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

2.4 SCOPE

This review was scoped to the Tenet repository.

1. IN-SCOPE TREE & COMMIT :

- evmos-update

Commit ID : a48990675a95cdb304cfcbbc32559f0514dd207b

REMEDIATION COMMIT IDs :

- c0d3388e33ff751e124c06c2cfab55ab83a29596
- d387f2d239a09cd9db10be877c0205ae2de49afa
- daffe0f2e481a166cbc9027718721b00df27ea89

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	4	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) HARDCODED GAS VALUE IN ALLIANCE STAKING - STAKING FUNCTIONS	Medium (6.2)	SOLVED - 11/01/2023
(HAL-02) UNCHECKED RETURN VALUE OF GETVALIDATOR	Medium (6.2)	SOLVED - 11/01/2023
(HAL-03) QUERIES DO NOT SUPPORT PAGINATION, ENABLING ATTACKERS TO DOS THE CHAIN	Low (3.8)	SOLVED - 11/01/2023
(HAL-04) GO VERSIONS PRIOR TO 1.20.2 CONTAIN CRYPTOGRAPHIC ISSUES AND OTHER BUGS	Low (2.3)	SOLVED - 11/01/2023
(HAL-05) LACK OF SPEC ON THE MODULE	Low (2.1)	SOLVED - 11/01/2023
(HAL-06) DOCKER IMAGE RUNNING AS ROOT	Low (3.8)	SOLVED - 11/01/2023
(HAL-07) USE OF DEPRECATED GO VERSION	Informational (0.0)	ACKNOWLEDGED
(HAL-08) PANIC IS USED FOR ERROR HANDLING	Informational (0.0)	ACKNOWLEDGED
(HAL-09) PRESENCE OF TO-DO COMMENTS ON THE CODE	Informational (0.0)	ACKNOWLEDGED
(HAL-10) LACK OF AUTHORIZATION (AUTHZ) GRANT IN DELEGATION METHOD	Informational (0.0)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) HARDCODED GAS VALUE IN ALLIANCE STAKING - STAKING FUNCTIONS - MEDIUM (6.2)

Description:

Within the alliance_staking package, the DelegationsHandler struct's Gas() method is set to return a hardcoded gas value of 50000. While this provides a constant gas estimation for operations related to delegations, it might not be the most efficient or accurate approach. Hardcoding such values can be problematic for several reasons:

- Hardcoding such values can lead to inefficiencies. If the actual gas required for the execution of the contract is more than the hardcoded value, it could result in out-of-gas errors. Conversely, if the actual gas required is significantly less than the hardcoded value, users might be overpaying for the execution, leading to inefficiencies.
- Users could be overpaying for gas if the actual requirement is much lower than the hardcoded value. This could deter users from interacting with the contract due to higher costs.

Note : The issue exists on the all other functionalities.

Code Location:

delegations.go#L26C1-L28C2

Listing 1

```
1 func (h DelegationsHandler) Gas() uint64 {
2     return 50000
3 }
```

BVSS:

AO:A/AC:L/AX:L/C:M/I:M/A:N/D:N/Y:N/R:N/S:U (6.2)

Recommendation:

Implement a mechanism to estimate the gas required for executing the functions based on its complexity and the operations it performs. Consider introducing a mechanism that allows administrators or the contract itself to adjust the gas value based on network conditions, contract updates, or other relevant factors.

Remediation Plan:

SOLVED: The Tenet team solved the issue by implementing dynamic gas calculation.

Commit ID: d387f2d239a09cd9db10be877c0205ae2de49afa

4.2 (HAL-02) UNCHECKED RETURN VALUE OF GETVALIDATOR - MEDIUM (6.2)

Description:

In the function Run of MintedBasecoinsHandler, the method GetValidator is called to retrieve validator details for a given address. The returned error value from GetValidator is ignored and not checked for potential issues.

If GetValidator encounters an error, such as the validator not being found (types.ErrNoValidatorFound) or any other internal error, the error is silently ignored. This can lead to subsequent operations in the Run function using uninitialized or incorrect data from the validator variable.

Code Location:

/precompiles/alliance_staking/minted_basecoins.go#L65

Listing 2

```
1 func (h MintedBasecoinsHandler) Run(evm *vm.EVM, contract *vm.
↳ Contract, readonly bool) ([]byte, error) {
2     ctx := evm.StateDB.(*statedb.StateDB).GetContext()
3
4     moduleAddr := h.accountKeeper.GetModuleAddress(alliancetypes.
↳ ModuleName)
5
6     amount := big.NewInt(0)
7
8     delegations := h.sk.GetAllDelegatorDelegations(ctx, moduleAddr
↳ )
9     for _, delegation := range delegations {
10        validator, _ := h.sk.GetValidator(ctx, delegation.
↳ GetValidatorAddr())
11        amount.Add(amount, validator.TokensFromShares(delegation.
↳ Shares).TruncateInt().BigInt())
12    }
```

```
13
14      return h.Method().Outputs.Pack(amount)
15  }
```

BVSS:

AO:A/AC:L/AX:L/C:M/I:M/A:N/D:N/Y:N/R:N/S:U (6.2)

Recommendation:

Consider checking the error return value of functions.

Remediation Plan:

SOLVED: The Tenet team solved the issue by checking address on the previous lines of the function.

4.3 (HAL-03) QUERIES DO NOT SUPPORT PAGINATION, ENABLING ATTACKERS TO DOS THE CHAIN - LOW (3.8)

Description:

The query implementations do not support result pagination. This could be problematic since some implemented Cosmos SDK queries could return a large number of items.

Additionally, gas is not charged depending on the size of the query result size, but rather on the query input size.

This could allow malicious actors to execute computationally and memory-heavy queries with a disproportionate gas cost, which could slow down block production up to the point where the chain halts.

Code Location:

minted_basecoins.go#L63-L64

Listing 3

```
func (h MintedBasecoinsHandler) Run(evm *vm.EVM, contract *vm.Contract, readonly bool) ([]byte, error) {
    ctx := evm.StateDB.(*statedb.StateDB).GetContext()

    moduleAddr := h.accountKeeper.GetModuleAddress(alliancetypes.ModuleName)

    amount := big.NewInt(0)

    delegations := h.sk.GetAllDelegatorDelegations(ctx, moduleAddr)
    for _, delegation := range delegations {
        validator, _ := h.sk.GetValidator(ctx, delegation.GetValidatorAddr())
        amount.Add(amount, validator.TokensFromShares(delegation.
```

```
↳ Shares).TruncateInt().BigInt())
12      }
13
14      return h.Method().Outputs.Pack(amount)
15  }
16
```

total_delegation.go#L57-L58

Listing 4

```
 1  func (h TotalDelegationHandler) Run(evm *vm.EVM, contract *vm.
↳ Contract, readonly bool) ([]byte, error) {
 2      ctx := evm.StateDB.(*statedb.StateDB).GetContext()
 3
 4      data := struct {
 5          Delegator common.Address
 6          Token     common.Address
 7      }{}
 8
 9      err := h.UnpackData(&data, contract.Input)
10      if err != nil {
11          return nil, err
12      }
13
14      amount := big.NewInt(0)
15
16      id := h.ek.GetERC20Map(ctx, data.Token)
17      if len(id) == 0 {
18          return nil, errors.New("Coin pair not found")
19      }
20
21      pair, found := h.ek.GetTokenPair(ctx, id)
22      if !found {
23          return nil, errors.New("Coin pair not found")
24      }
25
26      queryServer := alliancemodulekeeper.QueryServer{Keeper: h.ak}
27
28      resp, err := queryServer.AlliancesDelegation(ctx, &
↳ alliancetypes.QueryAlliancesDelegationsRequest{
29          DelegatorAddr: sdk.AccAddress(data.Delegator.Bytes()).
↳ String(),
```

```
30      })
31      if err != nil {
32          return nil, err
33      }
34
35      for _, delegation := range resp.Delegations {
36          if pair.Denom == delegation.Balance.Denom {
37              amount = amount.Add(amount, delegation.Balance.Amount.
↳ BigInt())
38          }
39      }
40
41      return h.Method().Outputs.Pack(amount)
42  }
```

BVSS:

AO:A/AC:L/AX:M/C:M/I:L/A:N/D:N/Y:N/R:N/S:U (3.8)

Recommendation:

It is recommended implementing pagination functionality and reasonable limits for queries.

Remediation Plan:

SOLVED: The Tenet team solved the issue by implementing dynamic gas calculation.

Commit ID: d387f2d239a09cd9db10be877c0205ae2de49afa

4.4 (HAL-04) GO VERSIONS PRIOR TO 1.20.2 CONTAIN CRYPTOGRAPHIC ISSUES AND OTHER BUGS - LOW (2.3)

Description:

Go version 1.20.2 contains security and performance enhancements. Specifically, this release fixes problems in cryptographic libraries. Older versions of go are more susceptible to cryptography issues and side-channel attacks on cryptographic implementations.

Code Location:

go.mod

Listing 5

```
1 module github.com/tenet-org/tenet-node
2
3 go 1.19
```

BVSS:

AO:A/AC:L/AX:L/C:L/I:L/A:M/D:M/Y:N/R:F/S:C (2.3)

Recommendation:

Update to Go v1.20.2 or newer when possible. More information can be found in the Go release notes.

Remediation Plan:

SOLVED: The Tenet team solved the issue by removing the docker file.

Commit ID: c0d3388e33ff751e124c06c2cfab55ab83a29596

4.5 (HAL-05) LACK OF SPEC ON THE MODULE - LOW (2.1)

Description:

The spec file is intended to outline the common structure for the specifications within this directory. Specifications are missing from **vesting** module. This documentation is segmented into messages focused on the developer and messages directed at the end user. These messages can be displayed to the end user (the human) at the time they will interact with the module.

Code Location:

- EVM Alliance Module
- Signed Blocks Counter Module

BVSS:

AO:A/AC:M/AX:L/C:N/I:M/A:N/D:N/Y:N/R:P/S:C (2.1)

Recommendation:

It is recommended that modules be fully annotated using specifications for all available functionality.

Remediation Plan:

SOLVED: The Tenet team solved the issue by adding specs on the modules.

Commit ID: daffe0f2e481a166cbc9027718721b00df27ea89

4.6 (HAL-06) DOCKER IMAGE RUNNING AS ROOT - LOW (3.8)

Description:

Docker containers generally run with root privileges by default. This allows for unrestricted container management, meaning a user could install system packages, edit configuration files, bind privileged ports, etc. During static analysis, it was observed that the docker image is maintained through the root user.

Code Location:

Listing 6: Dockerfile

```
1  FROM golang:1.19.5-bullseye AS build-env
2
3  WORKDIR /go/src/github.com/tenet-org/tenet-node
4
5  RUN apt-get update -y
6  RUN apt-get install git -y
7
8  COPY . .
9
10 RUN make build
11
12 FROM golang:1.19.5-bullseye
13
14 RUN apt-get update -y
15 RUN apt-get install ca-certificates jq -y
16
17 WORKDIR /root
18
19 COPY --from=build-env /go/src/github.com/tenet-org/tenet-node/
↳ build/tenetd /usr/bin/tenetd
20
21 EXPOSE 26656 26657 1317 9090 8545 8546
22
23 CMD ["tenetd"]
```

BVSS:

AO:A/AC:L/AX:M/C:M/I:L/A:N/D:N/Y:N/R:N/S:U (3.8)

Recommendation:

It is recommended to build the Dockerfile and run the container as a non-root user.

Listing 7: Reference

```
1 USER 1001: this is a non-root user UID, and here it is assigned to
↳  the image to run the current container as an unprivileged user.
↳ By doing so, the added security and other restrictions mentioned
↳ above are applied to the container.
```

Remediation Plan:

SOLVED: The Tenet team solved the issue by removing the docker file.

Commit ID: c0d3388e33ff751e124c06c2cfab55ab83a29596

4.7 (HAL-07) USE OF DEPRECATED GO VERSION - INFORMATIONAL (0.0)

Description:

The Docker environments used by Tenet use Go version 1.19. This version has been deprecated. See the Go release notes for their policy on supporting major versions of Go.

Code Location:

Dockerfile

Listing 8

```
1  FROM golang:1.19.5-bullseye AS build-env
2
3  WORKDIR /go/src/github.com/tenet-org/tenet-node
4
5  RUN apt-get update -y
6  RUN apt-get install git -y
7
8  COPY . .
9
10 RUN make build
11
12 FROM golang:1.19.5-bullseye
13
14 RUN apt-get update -y
15 RUN apt-get install ca-certificates jq -y
16
17 WORKDIR /root
18
19 COPY --from=build-env /go/src/github.com/tenet-org/tenet-node/
↳ build/tenetd /usr/bin/tenetd
20
21 EXPOSE 26656 26657 1317 9090 8545 8546
22
23 CMD ["tenetd"]
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:F/S:C (0.0)

Recommendation:

Update to a supported version of Go in order to receive ongoing security updates.

Remediation Plan:

ACKNOWLEDGED: The Tenet team acknowledged this finding.

4.8 (HAL-08) PANIC IS USED FOR ERROR HANDLING - INFORMATIONAL (0.0)

Description:

Multiple instances of the panic function have been identified in the codebase. They seem 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 affect the availability of the software to users.

Code Location:

Listing 9: Instances of panic identified in the codebase

```
./x/mint/keeper/keeper.go:31:       panic("the mint module account has not been set")
./x/mint/abci.go:28:        panic(err)
./x/mint/module.go:66:      panic(err)
./x/evm_alliance/keeper/keeper.go:53:       panic(err)
./x/evm_alliance/keeper/delegations.go:61:              panic(err)
./x/evm_alliance/abci.go:19:            panic(fmt.Errorf("failed to complete undelegations from x/evm_alliance module: %s", err))
./x/evm_alliance/genesis.go:35:     panic(fmt.Errorf("error setting params %s", err))
./x/evm_alliance/module.go:90:      panic(err)
./x/signed_blocks_counter/genesis_test.go:30:       panic(err)
./x/signed_blocks_counter/abci.go:27:               panic("Validator not found")
./x/signed_blocks_counter/genesis.go:19:            panic(err)
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Instead of using panics, custom errors should be defined and handled according to Best practices.

Remediation Plan:

ACKNOWLEDGED: The Tenet team acknowledged this finding.

4.9 (HAL-09) PRESENCE OF TO-DO COMMENTS ON THE CODE - INFORMATIONAL (0.0)

Description:

Multiple TO-DO comments were found on the code. From the security perspective, the use of these comments does not imply a security risk. However, It could mean that the developed contracts did not reach an appropriate level of maturity to be in a production environment.

Code Location:

Listing 10

```
./cmd/tenetd/main.go:51:    // TODO fix
./app/tps_counter.go:129:       // TODO: Perhaps log this?
./app/app.go:978:       // TODO: Record the count along with the
↳ code and or reason so as to display
./app/upgrades/tenet-2/upgrades.go:132: // todo: set proper
↳ address
./app/upgrades/tenet-2/upgrades.go:144: // todo: set proper
↳ address
./precompiles/suite_test.go:145:    // TODO change to setup with 1
↳  validator
./Makefile:146: # TODO replace with kaniko
./Makefile:452:# TODO: Rethink API docs generation
./buf.gen.proto.yaml:18:      - ./docs/protodoc-markdown.tmpl,
↳ proto-docs.md
./.golangci.yml:21:    # - lll TODO: enable
./.gitleaks.toml:705:    "todomvc",
./.gitleaks.toml:1208:    "todo",
./client/docs/swagger-ui/swagger-ui-bundle.js:18012:
↳ mapstodown: "",
./x/mint/types/expected_keepers.go:12:  // TODO remove with
↳ genesis 2-phases refactor https://github.com/cosmos/cosmos-sdk/
↳ issues/2862
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Review all the comments on the code and ensure that this situation does not affect or offer any risk to the security of the codebase.

Remediation Plan:

ACKNOWLEDGED: The Tenet team acknowledged this finding.

4.10 (HAL-10) LACK OF AUTHORIZATION (AUTHZ) GRANT IN DELEGATION METHOD - INFORMATIONAL (0.0)

Description:

The DelegateHandler method in the staking package does not support the Authorization (Authz) Grant that is prevalent in EVMOS. In the EVMOS's Delegate function, there's a clear mechanism to check for stake authorization and the allowance for the delegator. The lack of Authz also means that the system doesn't support scenarios where one entity is given permission to delegate on behalf of another entity, reducing flexibility for users.

Code Location:

delegate.go#L52-L53

Listing 11

```
func (h DelegateHandler) Run(evm *vm.EVM, contract *vm.Contract,
↳ readonly bool) ([]byte, error) {
    ctx := evm.StateDB.(*statedb.StateDB).GetContext()

    data := struct {
        Validator common.Address
    }{}

    err := h.UnpackData(&data, contract.Input)
    if err != nil {
        return nil, err
    }

    validator, found := h.sk.GetValidator(ctx, data.Validator.
↳ Bytes())
    if !found {
        return nil, errors.New("Validator not found")
    }

```

```
18      bondDenom := h.sk.BondDenom(ctx)
19      evm.StateDB.SubBalance(types.StakingAddress, contract.Value())
20
21      err = h.bk.MintCoins(ctx, evmtypes.ModuleName, sdk.NewCoins(
↳ sdk.NewCoin(bondDenom, sdk.NewIntFromBigInt(contract.Value()))))
22      if err != nil {
23          return nil, err
24      }
25
26      err = h.bk.SendCoinsFromModuleToAccount(ctx, evmtypes.
↳ ModuleName, contract.Caller().Bytes(), sdk.NewCoins(sdk.NewCoin(
↳ bondDenom, sdk.NewIntFromBigInt(contract.Value()))))
27      if err != nil {
28          return nil, err
29      }
30
31      shares, err := h.sk.Delegate(ctx, contract.Caller().Bytes(),
↳ sdk.NewIntFromBigInt(contract.Value()), stakingtypes.Unbonded,
↳ validator, true)
32      if err != nil {
33          return nil, err
34      }
35
36      return h.Method().Outputs.Pack(shares.BigInt())
37  }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Integrate the Authz grant mechanism into the DelegateHandler method, similar to how EVMOS handles it.

Remediation Plan:

ACKNOWLEDGED: The Tenet team acknowledged this finding.

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.

StaticCheck - Analysis Output Sample:

Listing 12

```
x/mint/simulation/params_test.go:25:29: undefined: simulation.ParamChanges (compile)
-: # github.com/tenet-org/tenet-node/x/mint_test [github.com/tenet-org/tenet-node/x/mint.test]
x/mint/module_test.go:21:121: undefined: simapp.EmptyAppOptions
x/mint/module_test.go:23:25: undefined: simapp.GenesisStateWithSingleValidator (compile)
contracts/compiled_contract.go:28:2: should replace this if statement with an unconditional strings.TrimPrefix (S1017)
x/evm_alliance/types/query.pb.gw.go:16:2: "github.com/golang/protobuf/descriptor" is deprecated: See the "google.golang.org/protobuf/reflect/protoreflect" package for how to obtain an EnumDescriptor or MessageDescriptor in order to programatically interact with the protobuf type system.  (SA1019)
x/evm_alliance/types/query.pb.gw.go:17:2: "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.org/protobuf/proto" package instead.  (SA1019)
x/evm_alliance/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 details.  (SA1019)
x/mint/module.go:143:70: simtypes.WeightedProposalContent is deprecated: Use WeightedProposalMsg instead.  (SA1019)
x/mint/types/query.pb.gw.go:16:2: "github.com/golang/protobuf/descriptor" is deprecated: See the "google.golang.org/protobuf/reflect/protoreflect" package for how to obtain an EnumDescriptor or MessageDescriptor in order to programatically interact with the protobuf type system.  (SA1019)
x/mint/types/query.pb.gw.go:17:2: "github.com/golang/protobuf/proto" is deprecated: Use the "google.golang.org/protobuf/proto" package instead.  (SA1019)
x/mint/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 details.  (SA1019)
x/signed_blocks_counter/module.go:131:73: simtypes.WeightedProposalContent is deprecated: Use WeightedProposalMsg
```

```
↳ instead.  (SA1019)
14 x/signed_blocks_counter/module.go:132:11: simtypes.
↳ WeightedProposalContent is deprecated: Use WeightedProposalMsg
↳ instead.  (SA1019)
```

Semgrep - Security Analysis Output Sample:

Command :

Listing 13

```
1 semgrep --config configFile
```

Output :

Listing 14

```
1
2
3
4  Scan Status
5
6   Scanning 64 files tracked by git with 1099 Code rules:
7
8   Language      Rules   Files          Origin      Rules
9
10  <multilang>      55     118          Community    1099
11  go               80      50
12
13   100% 0:00:00
14
15
16
17 1 Code Finding
18
19
20    mint/simulation/genesis.go
21       go.lang.security.audit.crypto.math_random.math-random-used
22          Do not use `math/rand`. Use `crypto/rand` instead.
23          Details: https://sg.run/6nK6
24
```

AUTOMATED TESTING

```
25              Autofix  crypto/rand
26              8 "math/rand"
27
```

THANK YOU FOR CHOOSING

// HALBORN