// HALBORN

Rarimo - Threshold Signature Module

Golang Security Assessment

Prepared by: **Halborn**

Date of Engagement: **July 3rd, 2023 - August 24th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/23/2023	Alejandro Taibo
0.2	Document Updates	08/24/2023	Alejandro Taibo
0.3	Draft Review	08/24/2023	Gokberk Gulgun
0.4	Draft Review	08/25/2023	Gabi Urrutia
1.0	Remediation Plan	09/11/2023	Alejandro Taibo
1.1	Remediation Plan Review	09/11/2023	Gokberk Gulgun
1.2	Remediation Plan Review	09/12/2023	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
Alejandro Taibo	Halborn	Alejandro.Taibo@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Rarimo engaged Halborn to conduct a security assessment on their module beginning on July 3rd, 2023 and ending on August 24th, 2023. The security assessment was scoped to the module provided to the Halborn team.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided seven weeks for the engagement and assigned one full-time security engineer to assess the security of the module implementation. The security engineers are blockchain and smart-contract security experts who are skilled in advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that the module implementation functions as intended.
- Identify potential security issues with the codebase.

In summary, Halborn identified some major security risks that were addressed by the Rarimo team along with minor improvements.

1.3 SCOPE

IN-SCOPE CODE & COMMIT:

- Repository: tss-svc
  - Commit ID: 05869cd940453f730b795b0033a2946b5f409b5e

REMEDIATION COMMITS:

- Repository: tss-svc
  - Merge request: merge_requests/16
  - Commit IDs:
    - 8d51feb03017c82bcbd353d1c6aa5fd0b579baf8
    - 719f5fe4a6966c0c6dbc9c2c3d342ca2821c8253
    - 595cfaecd90e78adf0dad09c81985ed374b750af
    - 5165631be6d452a6c6d458638853b64c2e1e4703
    - 76fe106cba720463bd6c391bb21ac56842de2377
    - f8b6abffb3f397016557b2eaa59e90b4d766192e
    - 7261907ed104c764ef64c5de05d0b2ffa299724b
    - b9a3542e60aab6cd715f426de13769cff7ac37a1

1.4 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 project.
- Custom fuzz testing using Go's built-in fuzzing tools.

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

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	1	2	4	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) A PARTY CAN MANIPULATE OTHER PARTIES TO REPORT ARBITRARY PROPOSERS AS OFFENDERS	Critical (10)	SOLVED - 09/11/2023
(HAL-02) NO SIGNER IS INTERNALLY SET IF SIGNACCEPTANCES IS GREATER THAN THRESHOLD	High (8.1)	SOLVED - 09/11/2023
(HAL-03) A PROPOSER IS NOT PUNISHED IF IT DOES NOT PROPOSE ANYTHING	Medium (6.2)	RISK ACCEPTED
(HAL-04) GRPC SERVER DOES NOT USE SSL/TLS	Medium (5.0)	SOLVED - 09/11/2023
(HAL-05) AUTH FUNCTION DOES NOT CHECK FOR MALLEABLE SIGNATURES	Low (3.8)	SOLVED - 09/11/2023
(HAL-06) DOCKER IMAGE RUNNING AS ROOT	Low (3.8)	RISK ACCEPTED
(HAL-07) FINISH CONTROLLER IS NOT HANDLING ERRORS WHEN TRY TO RETURN INDEXES TO THE POOL	Low (2.5)	SOLVED - 09/11/2023
(HAL-08) GRPC SUBMITTER DOES NOT USE SSL/TLS BY DEFAULT	Low (2.5)	RISK ACCEPTED
(HAL-09) USE OF DEPENDENCIES WITH PUBLIC VULNERABILITIES	Informational (0.0)	NOT APPLICABLE
(HAL-10) DEPRECATED GO VERSION IS BEING USED	Informational (0.0)	SOLVED - 09/11/2023
(HAL-11) LACK OF EXTENSIVE TEST COVERAGE	Informational (0.0)	ACKNOWLEDGED
(HAL-12) SENSITIVE INFORMATION IN THE ENVIRONMENT VARIABLES	Informational (0.0)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) A PARTY CAN MANIPULATE OTHER PARTIES TO REPORT ARBITRARY PROPOSERS AS OFFENDERS - CRITICAL(10)

Description:

During sign and reshare sessions, a proposer is required in order to generate the required data to sign, in case of sign session, or to prepare a new keygen process in order to validate new parties into the protocol by providing a new set of parties. It's important to highlight that this data is signed and verified by each party in order to guarantee that the data has been shared to the right proposer.

In each session, a proposer is selected deterministically by a common function implemented in each party; thus each party is aware of the current proposer in every session.

Additionally, parties can report other parties that are misbehaving, for example in case a party is sending data to other parties when it's not the current proposer, causing the signer of this data to be reported as offender. In case a specific number of parties report a party, this one will be frozen and kicked from the protocol.

The issue is located during the ProposalController execution, the RequestAuthorizer.Auth method does not verify if the received signature from other parties belong to the current session ID; therefore a malicious node could replay previous proposals, from past proposers, along with its corresponding signature in order to make other parties to report as offender to the past proposer when it's been selected a different one in the current session.

Proof of Concept:

1. Let's suppose a malicious party waits one session, behaving correctly at the same time it's storing the proposal received from another proposer during ProposalController execution.
2. When the session ends and a new one starts, the malicious party ensures that the new proposer is different from the one selected in the previous session.
3. During ProposalController execution in the new session, the malicious party broadcasts the previous proposal along with its signature that was shared in the latest session.
4. Each party will receive the replayed proposal. However, since the sender (obtained from the signature) is not equal to the current proposal, each party will report to the previous proposer as an offender.
5. Due to all parties will report to the previous proposer, it will be frozen and kicked out by the protocol.

Code Location:

Listing 1: internal/core/controllers/controller_proposal.go (Lines 44,49)

```
41 func (p *ProposalController) Receive(c context.Context, request *
↳ types.MsgSubmitRequest) error {
42     ctx := core.WrapCtx(c)
43
44     sender, err := p.auth.Auth(request)
45     if err != nil {
46         return err
47     }
48
49     if !Equal(sender, &p.data.Proposer) {
50         p.data.Offenders[sender.Account] = struct{}{}
51         return ErrSenderIsNotProposer
52     }
```

Listing 2: internal/core/auth.go

```
31 unc (r *RequestAuthorizer) Auth(request *types.MsgSubmitRequest)
↳ (*rarimo.Party, error) {
32     hash := crypto.Keccak256(request.Details.Value)
33
34     signature, err := hexutil.Decode(request.Signature)
35     if err != nil {
36         r.log.WithError(err).Debug("Failed to decode signature")
37         return nil, ErrInvalidSignature
38     }
39
40     pub, err := crypto.Ecrecover(hash, signature)
41     if err != nil {
42         r.log.WithError(err).Debug("Failed to recover signature
↳ pub key")
43         return nil, ErrInvalidSignature
44     }
45
46     // TODO optimize: make log(n)
47     key := hexutil.Encode(pub)
48     for _, p := range r.parties {
49         if p.PubKey == key {
50             return p, nil
51         }
52     }
53
54     return nil, ErrSignerNotAParty
55 }
```

BVSS:

AO:A/AC:L/AX:L/C:C/I:C/A:C/D:N/Y:N/R:N/S:U (10)

Recommendation:

It is recommended to verify when a signature was generated, for example by including the current session ID in the signature, or using a nonce in order to avoid replayable signatures.

Remediation Plan:

SOLVED: The Rarimo team solved the issue by including session ID and type into signed request data in the following commit:

- 595cfaecd90e78adf0dad09c81985ed374b750af.

4.2 (HAL-02) NO SIGNER IS INTERNALLY SET IF SIGNACCEPTANCES IS GREATER THAN THRESHOLD - HIGH (8.1)

Description:

During reshare session, new parties currently not verified using a proposal in order to take part of the protocol after the keygen process and the submission of the new key to the core including new parties.

In the first stage of this session, a proposer has to nominate the new set of parties that will take part of the protocol, consequently, this set has to be accepted by each party in the protocol, including the new ones. Once the proposal is accepted, each node has to verify if the set of parties is correct and determine which parties will act as a signer in the keygen process. Following the flow of the program, the reshareAcceptanceController.finish function is in charge of selecting deterministically the signed from an old set of parties. When len(signAcceptances)== a.data.Set.T+1, each node will try to find its address in an array of Signers in order to know if it has to be part of the keygen process.

It's well known that the amount of signers must reach threshold + 1 value to sign new data, but sometimes the number of signers is greater than threshold + 1 and parties have to select which ones will take part of the keygen process. However, the reshareAcceptanceController.finish function is not setting any signer after the set of signers has been defined; therefore, no party will be able to take part in keygen process when len(signAcceptances)> a.data.Set.T+1.

Code Location:

Listing 3: internal/core/controllers/controller_acceptance.go

```
248 func (a *reshareAcceptanceController) finish(ctx core.Context) {
249   if len(a.data.Acceptances) < a.data.Set.N {
250     a.data.Processing = false
251     return
252   }
253
254   defer func() {
255     ctx.Log().Infof("Session signers: %v", acceptancesToArr(a.data
 ↳ .Signers))
256   }()
257
258   signAcceptances := filterAcceptances(a.data.Acceptances, a.data.
 ↳ Set.VerifiedParties)
259
260   if len(signAcceptances) == a.data.Set.T+1 {
261     a.data.Signers = signAcceptances
262     _, a.data.IsSigner = a.data.Signers[ctx.SecretStorage().
 ↳ GetTssSecret().AccountAddress()]
263     return
264   }
265
266   a.data.Signers = GetSignersSet(signAcceptances, a.data.Set.T, a.
 ↳ data.Set.LastSignature, a.data.SessionId)
267 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:H/D:N/Y:N/R:N/S:U (8.1)

Recommendation:

This function should implement a defer function in charge of checking if the own party is part of the signer set.

Listing 4

```
1 defer func() {
2     ctx.Log().Infof("Session signers: %v", acceptancesToArr(a.data
↳ .Signers))
3     _, a.data.IsSigner = a.data.Signers[ctx.SecretStorage().
↳ GetTssSecret().AccountAddress()]
4   }()
```

Remediation Plan:

SOLVED: The Rarimo team solved the issue by including IsSigner assignation into a defer function in the following commit:

- 8d51feb03017c82bcbd353d1c6aa5fd0b579baf8.

4.3 (HAL-03) A PROPOSER IS NOT PUNISHED IF IT DOES NOT PROPOSE ANYTHING - MEDIUM (6.2)

Description:

As it was explained in previous issues, during each session a new proposer is selected deterministically. This proposer has to generate an expected data which will be sent and checked by each party in the protocol.

However, a proposer is able to not propose anything without getting a penalty due to this misbehavior since `d.data.Processing` flag will never be set to `true` and therefore, the `finish` controller will get executed without updating any data.

Code Location:

Listing 5: internal/core/controllers/controller_proposal.go

```
82 func (p *ProposalController) Next() IController {
83   if p.data.Processing {
84     return p.data.GetAcceptanceController()
85   }
86
87   return p.data.GetFinishController()
88 }
```

Listing 6: internal/core/controllers/controller_finish.go

```
135 func (d *defaultFinishController) finish(ctx core.Context) {
136   if d.data.Processing {
```

BVSS:

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

Recommendation:

It is recommended to add a mechanism to report as offenders to proposers that are not proposing anything during their turn.

Remediation Plan:

RISK ACCEPTED: The Rarimo team states that it is normal that proposer does not propose anything. Proposer observes current state of core and proposes only if some operations that should be signed exists.

4.4 (HAL-04) GRPC SERVER DOES NOT USE SSL/TLS - MEDIUM (5.0)

Description:

It's been identified that the GRPC server in charge of handling the request from other parties is not using SSL nor TLS. Therefore, not following the next statement specified in the Binance's TSS documentation:

When you build a transport, it should offer a broadcast channel as well as point-to-point channels connecting every pair of parties. Your transport should also employ suitable end-to-end encryption (TLS with an AEAD cipher is recommended)between parties to ensure that a party can only read the messages sent to it.

This issue should be fixed in order to avoid possible Man-in-The-Middle attacks.

Code Location:

Listing 7: internal/grpc/server_impl.go (Line 48)

```
47 func (s *ServerImpl) RunGRPC() error {
48   grpcServer := grpc.NewServer()
49   types.RegisterServiceServer(grpcServer, s)
50   return grpcServer.Serve(s.listener)
51 }
```

BVSS:

AO:A/AC:L/AX:L/C:M/I:N/A:N/D:N/Y:N/R:N/S:U (5.0)

Recommendation:

It is recommended to implement SSL/TLS by default in the GRPC server.

Listing 8

```
conf := &tls.Config{
    Certificates: []tls.Certificate{serverCert},
    ClientAuth:   tls.RequireAndVerifyClientCert,
    ClientCAs:    certPool,
}

tlsCredentials := credentials.NewTLS(conf)
grpcServer := grpc.NewServer(grpc.Creds(tlsCredentials))
```

Remediation Plan:

SOLVED: The Rarimo team solved the issue by ensuring that TLS connection will be applied on the top layer of launched system - load balancer or proxy.

4.5 (HAL-05) AUTH FUNCTION DOES NOT CHECK FOR MALLEABLE SIGNATURES - LOW (3.8)

Description:

The crypto.Ecrecover function is used in RequestAuthorizer.Auth to recover the public key from a signature included in a request. The crypto.Ecrecover function is susceptible to signature malleability, which could lead to replay attacks.

Code Location:

Listing 9: internal/core/auth.go (Line 40)

```
31 func (r *RequestAuthorizer) Auth(request *types.MsgSubmitRequest)
↳ (*rarimo.Party, error) {
32     hash := crypto.Keccak256(request.Details.Value)
33
34     signature, err := hexutil.Decode(request.Signature)
35     if err != nil {
36         r.log.WithError(err).Debug("Failed to decode signature")
37         return nil, ErrInvalidSignature
38     }
39
40     pub, err := crypto.Ecrecover(hash, signature)
41     if err != nil {
42         r.log.WithError(err).Debug("Failed to recover signature
↳ pub key")
43         return nil, ErrInvalidSignature
44     }
45
46     // TODO optimize: make log(n)
47     key := hexutil.Encode(pub)
48     for _, p := range r.parties {
49         if p.PubKey == key {
50             return p, nil
51         }
52     }
```

```
53
54     return nil, ErrSignerNotAParty
55 }
```

BVSS:

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

Recommendation:

The RequestAuthorizer.Auth function should implement a check in charge of discarding signatures whose value is greater than 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0 in order to prevent malleable signatures' usage. For this purpose, it is recommended to use EcdsaVerify.

Remediation Plan:

SOLVED: The Rarimo team solved the issue by using crypto.VerifySignature function from github.com/ethereum/go-ethereum/crypto which rejects malleable signatures in the following commit:

- 7261907ed104c764ef64c5de05d0b2ffa299724b.

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 10: Dockerfile

```
1 FROM golang:1.18-alpine as buildbase
2
3 RUN apk add git build-base
4
5 WORKDIR /go/src/gitlab.com/rarimo/tss/tss-svc
6 COPY vendor .
7 COPY . .
8
9 ENV GO111MODULE="on"
10 ENV CGO_ENABLED=1
11 ENV GOOS="linux"
12
13 RUN GOOS=linux go build  -o /usr/local/bin/tss-svc /go/src/gitlab.
↳ com/rarimo/tss/tss-svc
14
15
16 FROM alpine:3.9
17
18 COPY --from=buildbase /usr/local/bin/tss-svc /usr/local/bin/tss-
↳ svc
19 RUN apk add --no-cache ca-certificates
20
21 ENTRYPOINT ["tss-svc"]
```

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 11: 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:

RISK ACCEPTED: The Rarimo team states that Dockerfile from TSS repository is not used for stage or production builds, we’re using it only for local development.

4.7 (HAL-07) FINISH CONTROLLER IS NOT HANDLING ERRORS WHEN TRY TO RETURN INDEXES TO THE POOL - LOW (2.5)

Description:

When a session ends, operation indexes that were not signed or included in a signing process should be returned to the pool of unsigned operations. The defaultFinishController.returnToPool is in charge of this purpose.

However, when ctx.Pool().Add(index) is executed, no return value is handled. Therefore, in case an error occurs trying to add the unsigned operation to the pool, the program won't be able to handle it.

Code Location:

Listing 12: internal/core/controllers/controller_finish.go (Line 158)

```
155 func (d *defaultFinishController) returnToPool(ctx core.Context) {
156   // try to return indexes back to the pool
157   for _, index := range d.data.Indexes {
158     ctx.Pool().Add(index)
159   }
160 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:L/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

It is recommended to handle errors coming from ctx.Pool().Add(index) execution in order to avoid future issues.

Remediation Plan:

SOLVED: The Rarimo team solved the issue by adding error logging in the following commit:

- 719f5fe4a6966c0c6dbc9c2c3d342ca2821c8253.

4.8 (HAL-08) GRPC SUBMITTER DOES NOT USE SSL/TLS BY DEFAULT - LOW (2.5)

Description:

It has been identified that the program makes GRPC requests without SSL/TLS by default. However, it's recommended to use SSL/TLS by default instead of plain GRPC.

Code Location:

Listing 13: internal/connectors/submit.go (Line 73)

```
73   connectSecurityOptions := grpc.WithInsecure()
74
75   if s.secret.TLS() {
76     tlsConfig := &tls.Config{
77       InsecureSkipVerify: true,
78     }
79
80     connectSecurityOptions = grpc.WithTransportCredentials(
↳ credentials.NewTLS(tlsConfig))
81   }
82
83   client, err = grpc.Dial(addr, connectSecurityOptions)
```

BVSS:

AO:A/AC:L/AX:L/C:L/I:N/A:N/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

It is recommended to use SSL/TLS and deprecate plain GRPC.

Remediation Plan:

RISK ACCEPTED: The Rarimo team states that they have to support insecure connection to be able to launch the full system on local machines.

4.9 (HAL-09) USE OF DEPENDENCIES WITH PUBLIC VULNERABILITIES - INFORMATIONAL (0.0)

Description:

Several external packages are outdated and/or contain known vulnerabilities.

Code Location:

Excerpts of the output from both tools can be found in the Automated Testing section at the end of the report.

BVSS:

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

Recommendation:

Where possible, keep dependencies patched in order to reduce the risk of the system being attacked using known vulnerabilities. A tool like govulncheck can be added to the project's CI pipeline. This can then be configured to show serious issues that could affect the project.

It is important to note that many of these vulnerabilities flagged by govulncheck are unlikely to be exploitable in practice, as they larger refer to a Web2 context. In addition, the nancy tool reported issues that Halborn determined to be false positives.

Halborn recommends running the nancy and govulncheck tools regularly and to fix as many warnings as possible.

Remediation Plan:

NOT APPLICABLE: The aforementioned tools flagged incorrectly the go version in use. However, the go version was bumped in the following commit:

- 76fe106cba720463bd6c391bb21ac56842de2377.

4.10 (HAL-10) DEPRECATED GO VERSION IS BEING USED - INFORMATIONAL (0.0)

Description:

The project uses Go version 1.18. This version has been deprecated. See the Go release notes for their policy on supporting major versions of Go.

Code Location:

go.mod

Listing 14

```
1 module gitlab.com/rarimo/tss/tss-svc
2
3 go 1.18
```

BVSS:

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

Recommendation:

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

Remediation Plan:

SOLVED: The Rarimo team solved the issue by bumping go version in the following commit:

- 76fe106cba720463bd6c391bb21ac56842de2377.

4.11 (HAL-11) LACK OF EXTENSIVE TEST COVERAGE - INFORMATIONAL (0.0)

Description:

Adequate test coverage and regular reporting is an essential process to ensure the codebase works as intended. Insufficient code coverage can lead to unexpected issues and regressions due to changes in service implementation.

Code Location:

Code Location

BVSS:

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

Recommendation:

Make sure that the coverage report produced via **go test -cover** covers all functions.

Remediation Plan:

ACKNOWLEDGED: The Rarimo team acknowledged the issue and states they will consider providing more unit tests in the near future.

4.12 (HAL-12) SENSITIVE INFORMATION IN THE ENVIRONMENT VARIABLES - INFORMATIONAL (0.0)

Description:

The provided configuration contains sensitive data, such as the private tokens. Storing these values in a plain-text configuration file or environment variables can make it easier for attackers to gain unauthorized access to the oracle.

The insecure storage and handling of sensitive configuration data on a single VPS server can lead to various negative consequences, including but not limited to:

- Sensitive data leakage: Exposure of sensitive information, such as API keys, private keys, and login credentials, can enable attackers to gain unauthorized access to the service and its associated resources. This can result in further unauthorized actions, such as data manipulation or theft, disruption of services, and reputational damage to the organization.
- Unauthorized access to the API: With access to the API keys and credentials, attackers can make unauthorized API calls and potentially gain access to sensitive data, manipulate data, or perform other malicious activities.
- Potential loss of assets: Exposure of private keys for blockchain contracts can lead to unauthorized transactions or manipulation of the contract, resulting in potential loss of assets or funds.
- Increased risk of server compromise: Storing sensitive data in plaintext on a VPS server increases the attack surface, making it more attractive for attackers to target the server. A successful compromise of the server can lead to further damage, such as the installation of malware, lateral movement within the infrastructure, or complete takeover of the server.

Code Location:

Listing 15: docker-compose.yaml

```
- POSTGRES_USER=tss
- POSTGRES_PASSWORD=tss
- POSTGRES_DB=tss
- PGDATA=/pgdata
```

Listing 16: internal/config/vault.go

```
func (c *config) Vault() *vault.KVv2 {
    return c.vault.Do(func() interface{} {
        conf := vault.DefaultConfig()
        conf.Address = os.Getenv(VaultPathEnv)

        client, err := vault.NewClient(conf)
        if err != nil {
            panic(err)
        }

        client.SetToken(os.Getenv(VaultTokenEnv))

        return client.KVv2(os.Getenv(VaultMountPath))
    }).(*vault.KVv2)
}
```

Listing 17: internal/secret/vault.go

```
func NewVaultStorage(cfg config.Config) *VaultStorage {
    return &VaultStorage{
        client: cfg.Vault(),
        log:    cfg.Log(),
        path:   os.Getenv(config.VaultSecretPath),
    }
}
```

BVSS:

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

Recommendation:

To mitigate the potential risks and secure the sensitive configuration data on the VPS server, the following steps are suggested:

- Store sensitive data securely using a secret manager like HashiCorp Vault or AWS Secrets Manager. Avoid using environment variables for sensitive data, and instead, retrieve them directly from the secret manager in the service code.
- Harden the VPS server by implementing security best practices, such as keeping the server up-to-date, disabling unnecessary services, and restricting access using firewall rules and strong authentication mechanisms.
- Segregate responsibilities by deploying separate servers or containers for different components of the service. For example, use a dedicated server or container for the API, another for the database, and another for the secret manager.
- Regularly monitor and assessment the VPS server for signs of intrusion or other security issues. Configure intrusion detection and prevention systems (IDPS) and implement centralized logging for better visibility and faster incident response.
- Enable encryption for data in transit and at rest to protect sensitive data from being intercepted or accessed by unauthorized users.

Remediation Plan:

ACKNOWLEDGED: The Rarimo team acknowledged the issue and states currently all sensitive data (like secret keys) is stored in the HashiCorp Vault. Of course, connection to the Vault requires authorization and authorization requires the credentials provided in some way in the environment. Recommendations will be shared with all validators to clear envs after service launch.

AUTOMATED TESTING

5.1 Description

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec, semgrep, codeQL, govulncheck 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.

5.2 Semgrep

Security Analysis Output Sample:

Listing 18: Rule Set

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

AUTOMATED TESTING

Semgrep Results:

```
Scan Status

Scanning 130 files with 1071 Code rules:

Language      Rules   Files          Origin      Rules
<multilang>      60     311          Community    1071
go               82      59
php              33      14
json              4      11
yaml             28       7
dockerfile        4       1
bash              4       1

                                    100% 0:00:00


19 Code Findings

  docker-compose.yaml
     yaml.docker-compose.security.no-new-privileges.no-new-privileges
        Service 'tss1' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-
        privileges:true' in 'security_opt' to prevent this.
        Details: https://sg.run/0n8q

          4┆ tss1:
           ┆----------------------------------------
     yaml.docker-compose.security.no-new-privileges.no-new-privileges
        Service 'tss1-db' allows for privilege escalation via setuid or setgid binaries. Add 'no-
        new-privileges:true' in 'security_opt' to prevent this.
        Details: https://sg.run/0n8q

         17┆ tss1-db:
           ┆----------------------------------------
     yaml.docker-compose.security.no-new-privileges.no-new-privileges
        Service 'tss2' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-
        privileges:true' in 'security_opt' to prevent this.
        Details: https://sg.run/0n8q

         31┆ tss2:
           ┆----------------------------------------
     yaml.docker-compose.security.no-new-privileges.no-new-privileges
        Service 'tss2-db' allows for privilege escalation via setuid or setgid binaries. Add 'no-
        new-privileges:true' in 'security_opt' to prevent this.
        Details: https://sg.run/0n8q

         44┆ tss2-db:
           ┆----------------------------------------
     yaml.docker-compose.security.no-new-privileges.no-new-privileges
        Service 'tss3' allows for privilege escalation via setuid or setgid binaries. Add 'no-new-
        privileges:true' in 'security_opt' to prevent this.
        Details: https://sg.run/0n8q

         58┆ tss3:
           ┆----------------------------------------
     yaml.docker-compose.security.no-new-privileges.no-new-privileges
        Service 'tss3-db' allows for privilege escalation via setuid or setgid binaries. Add 'no-
        new-privileges:true' in 'security_opt' to prevent this.
        Details: https://sg.run/0n8q

         71┆ tss3-db:
           ┆----------------------------------------
     yaml.docker-compose.security.writable-filesystem-service.writable-filesystem-service
        Service 'tss1' is running with a writable root filesystem. This may allow malicious
        applications to download and run additional payloads, or modify container files. If an
        application inside a container has to save something temporarily consider using a tmpfs. Add
        'read_only: true' to this service to prevent this.
        Details: https://sg.run/e4JE

          4┆ tss1:
           ┆----------------------------------------
     yaml.docker-compose.security.writable-filesystem-service.writable-filesystem-service
        Service 'tss1-db' is running with a writable root filesystem. This may allow malicious
        applications to download and run additional payloads, or modify container files. If an
        application inside a container has to save something temporarily consider using a tmpfs. Add
        'read_only: true' to this service to prevent this.
        Details: https://sg.run/e4JE

         17┆ tss1-db:
```

AUTOMATED TESTING

```
         71┆ tss3-db:

  internal/cli/main.go
     go.lang.security.audit.net.use-tls.use-tls
        Found an HTTP server without TLS. Use 'http.ListenAndServeTLS' instead. See
        https://golang.org/pkg/net/http/#ListenAndServeTLS for more information.
        Details: https://sg.run/dKbY

         ▶▶┆ Autofix ▶ http.ListenAndServeTLS(":8080", certFile, keyFile, r)
        158┆ if err := http.ListenAndServe(":8080", r); err != nil {

  internal/config/cosmos.go
     go.lang.security.audit.crypto.missing-ssl-minversion.missing-ssl-minversion
        `MinVersion` is missing from this TLS configuration.  By default, TLS 1.2 is currently used
        as the minimum when acting as a client, and TLS 1.0 when acting as a server. General purpose
        web applications should default to TLS 1.3 with all other protocols disabled.  Only where it
        is known that a web server must support legacy clients with unsupported an insecure browsers
        (such as Internet Explorer 10), it may be necessary to enable TLS 1.0 to provide support.
        Add `MinVersion: tls.VersionTLS13' to the TLS configuration to bump the minimum version to
        TLS 1.3.
        Details: https://sg.run/oxEN

         ▶▶┆ Autofix ▶ tls.Config{ InsecureSkipVerify: true, MinVersion:
tls.VersionTLS13 }
         34┆ tlsConfig := &tls.Config{
         35┆   InsecureSkipVerify:
true,
         36┆ }
          ┆----------------------------------------
     problem-based-packs.insecure-transport.go-stdlib.bypass-tls-verification.bypass-tls-
     verification
        Checks for disabling of TLS/SSL certificate verification. This should only be used for
        debugging purposes because it leads to vulnerability to MTM attacks.
        Details: https://sg.run/4xj5

         34┆ tlsConfig := &tls.Config{
         35┆   InsecureSkipVerify:
true,
         36┆ }

  internal/connectors/submit.go
     go.lang.security.audit.crypto.missing-ssl-minversion.missing-ssl-minversion
        `MinVersion` is missing from this TLS configuration.  By default, TLS 1.2 is currently used
        as the minimum when acting as a client, and TLS 1.0 when acting as a server. General purpose
        web applications should default to TLS 1.3 with all other protocols disabled.  Only where it
        is known that a web server must support legacy clients with unsupported an insecure browsers
        (such as Internet Explorer 10), it may be necessary to enable TLS 1.0 to provide support.
        Add `MinVersion: tls.VersionTLS13' to the TLS configuration to bump the minimum version to
        TLS 1.3.
        Details: https://sg.run/oxEN

         ▶▶┆ Autofix ▶ tls.Config{ InsecureSkipVerify: true, MinVersion:
tls.VersionTLS13 }
         76┆ tlsConfig := &tls.Config{
         77┆   InsecureSkipVerify:
true,
         78┆ }
          ┆----------------------------------------
     problem-based-packs.insecure-transport.go-stdlib.bypass-tls-verification.bypass-tls-
     verification
```

```
        Checks for disabling of TLS/SSL certificate verification. This should only be used for
        debugging purposes because it leads to vulnerability to MTM attacks.
        Details: https://sg.run/4xj5

         76┆ tlsConfig := &tls.Config{
         77┆   InsecureSkipVerify:
true,
         78┆ }

  internal/grpc/server_impl.go
     go.grpc.security.grpc-server-insecure-connection.grpc-server-insecure-connection
        Found an insecure gRPC server without 'grpc.Creds()' or options with credentials. This
        allows for a connection without encryption to this server. A malicious attacker could tamper
        with the gRPC message, which could compromise the machine. Include credentials derived from
        an SSL certificate in order to create a secure gRPC connection. You can create credentials
        using 'credentials.NewServerTLSFromFile("cert.pem", "cert.key")'.
        Details: https://sg.run/5Q5l

         48┆ grpcServer := grpc.NewServer()
           ┆----------------------------------------
     go.lang.security.audit.net.use-tls.use-tls
        Found an HTTP server without TLS. Use 'http.ListenAndServeTLS' instead. See
        https://golang.org/pkg/net/http/#ListenAndServeTLS for more information.
        Details: https://sg.run/dKbY

         ▶▶┆ Autofix ▶ http.ListenAndServeTLS(s.swagger.Addr, certFile, keyFile,
httpRouter)
         69┆ return http.ListenAndServe(s.swagger.Addr, httpRouter)
```

- No major issues found by Semgrep.

5.3 Gosec

Analysis Output Sample:

```
[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/connectors/submit_go:77] - G402 (CWE-295): TLS InsecureSkipVerify set true. (Confidence: HIGH, Severity: HIGH)
   76:                 tlsConfig := &tls.Config{
  > 77:                         InsecureSkipVerify: true,
   78:                 }


[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/config/cosmos.go:35]
 - G402 (CWE-295): TLS InsecureSkipVerify set true. (Confidence: HIGH, Severity: HIGH)
   34:                         tlsConfig := &tls.Config{
  > 35:                                 InsecureSkipVerify: true,
   36:                         }


[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/grpc/server_impl.go:69] - G114 (CWE-676): Use of net/http serve function that has no support for setting timeouts (Confidence: H
IGH, Severity: MEDIUM)
   68:         httpRouter.Handle("/", grpcGatewayRouter)
  > 69:         return http.ListenAndServe(s.swagger.Addr, httpRouter)
   70: }


[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/cli/main.go:158] - G114 (CWE-676): Use of net/http serve function that has no support for setting timeouts (Confidence: HIGH, Se
verity: MEDIUM)
   157:
  > 158:        if err := http.ListenAndServe(":8080", r); err != nil {
   159:                panic(err)


[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/secret/util.go:60] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   59:
  > 60:         data, err := os.ReadFile(path)
   61:         if err != nil {


[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/secret/util.go:40] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   39:
  > 40:         data, err := os.ReadFile(path)
   41:         if err != nil {


[/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/core/controllers/controller_finish.go:158] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
   157:        for _, index := range d.data.Indexes {
  > 158:                ctx.Pool().Add(index)
   159:        }


Summary:
  Gosec  : 2.16.0
  Files  : 59
  Lines  : 8435
  Nosec  : 0
  Issues : 7
```

- No major issues found by Gosec.

5.4 StaticCheck

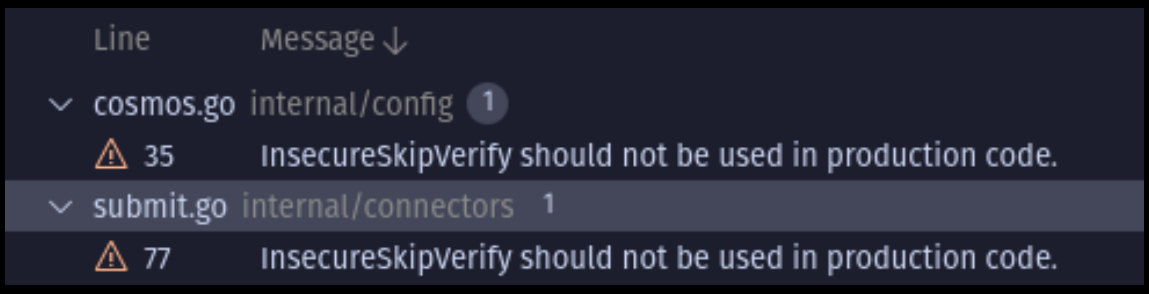
Analysis Output Sample:

```
/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/config/cosmos.go
  (31, 29)  SA1019  grpc.WithInsecure is deprecated: use WithTransportCredentials and insecure.NewCredentials() instead. Will be supported throughout 1.x.

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/config/main.go
  (36, 2)  U1000  field private is unused

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/config/tendermint.go
  (4, 2)    ST1019  package "github.com/tendermint/tendermint/rpc/client/http" is being imported more than once
    (5, 2)            other import of "github.com/tendermint/tendermint/rpc/client/http"

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/connectors/submit.go
  (73, 28)  SA1019  grpc.WithInsecure is deprecated: use WithTransportCredentials and insecure.NewCredentials() instead. Will be supported throughout 1.x.

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/core/bounds.go
  (35, 2)  U1000  field mu is unused

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/core/controllers/controller_signature.go
  (211, 3)  SA4006  this value of err is never used
  (233, 3)  SA4006  this value of err is never used

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/core/controllers/util.go
  (257, 9)  S1005  unnecessary assignment to the blank identifier

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/secret/util.go
  (54, 6)  U1000  func loadData is unused
  (70, 6)  U1000  func saveData is unused

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/timer/subscriber.go
  (61, 4)  S1023  redundant break statement

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/tss/keygen.go
  (195, 5)  S1005  unnecessary assignment to the blank identifier

/home/kaorz/Documents/Work/Halborn/Projects/rarimo/tss-last-commit/tss-svc/internal/tss/sign.go
  (208, 5)  S1005  unnecessary assignment to the blank identifier

✖ 13 problems (13 errors, 0 warnings, 0 ignored)
```

- No major issues found by StaticCheck.

5.5 CodeQL

Analysis Output Sample (go queries):



- No major issues found by CodeQL.

5.6 Govulncheck

Analysis Output Sample:

```
Vulnerability #1: GO-2023-1987
  Extremely large RSA keys in certificate chains can cause a
  client/server to expend significant CPU time verifying
  signatures. With fix, the size of RSA keys transmitted during
  handshakes is restricted to <= 8192 bits. Based on a survey of
  publicly trusted RSA keys, there are currently only three
  certificates in circulation with keys larger than this, and all
  three appear to be test certificates that are not actively
  deployed. It is possible there are larger keys in use in private
  PKIs, but we target the web PKI, so causing breakage here in the
  interests of increasing the default safety of users of
  crypto/tls seems reasonable.

  More info: https://pkg.go.dev/vuln/GO-2023-1987

  Standard library
    Found in: crypto/tls@go1.20.3
    Fixed in: crypto/tls@go1.21rc4

    Call stacks in your code:
      internal/config/cosmos.go:18:20: gitlab.com/rarimo/tss/tss-svc/internal/config.config.Cosmos calls gitlab.com/rarimo/tss/tss-svc/internal/config.Tendermint, which eventually calls crypto/tls.Conn.HandshakeCo
ntext
      internal/grpc/server_impl.go:50:25: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGRPC calls google.golang.org/grpc.Server.Serve, which eventually calls crypto/tls.Conn.Write
      internal/secret/vault.go:77:32: gitlab.com/rarimo/tss/tss-svc/internal/secret.VaultStorage.SetTssSecret calls github.com/hashicorp/vault/api.KVv2.Put, which eventually calls crypto/tls.Conn.Read
      internal/timer/main.go:21:32: gitlab.com/rarimo/tss/tss-svc/internal/timer.NewTimer calls github.com/tendermint/tendermint/rpc/client/http.baseRPCClient.Status, which eventually calls crypto/tls.Dialer.DialC
ontext

Vulnerability #2: GO-2023-1878
  The HTTP/1 client does not fully validate the contents of the
  Host header. A maliciously crafted Host header can inject
  additional headers or entire requests. With fix, the HTTP/1
  client now refuses to send requests containing an invalid
  Request.Host or Request.URL.Host value.

  More info: https://pkg.go.dev/vuln/GO-2023-1878

  Standard library
    Found in: net/http@go1.20.3
    Fixed in: net/http@go1.20.6

    Call stacks in your code:
      internal/config/cosmos.go:18:20: gitlab.com/rarimo/tss/tss-svc/internal/config.config.Cosmos calls gitlab.com/rarimo/tss/tss-svc/internal/config.Tendermint, which eventually calls net/http.Request.Write
      internal/secret/vault.go:77:32: gitlab.com/rarimo/tss/tss-svc/internal/secret.VaultStorage.SetTssSecret calls github.com/hashicorp/vault/api.KVv2.Put, which eventually calls net/http.Client.CloseIdleConnecti
ons
      internal/timer/main.go:21:32: gitlab.com/rarimo/tss/tss-svc/internal/timer.NewTimer calls github.com/tendermint/tendermint/rpc/client/http.baseRPCClient.Status, which eventually calls net/http.Client.Do

Vulnerability #3: GO-2023-1840
  On Unix platforms, the Go runtime does not behave differently
  when a binary is run with the setuid/setgid bits. This can be
  dangerous in certain cases, such as when dumping memory state,
  or assuming the status of standard i/o file descriptors. If a
  setuid/setgid binary is executed with standard I/O file
  descriptors closed, opening any files can result in unexpected
  content being read or written with elevated privileges.
  Similarly, if a setuid/setgid program is terminated, either via
  panic or signal, it may leak the contents of its registers.

  More info: https://pkg.go.dev/vuln/GO-2023-1840

  Standard library
    Found in: runtime@go1.20.3
    Fixed in: runtime@go1.20.5

    Call stacks in your code:
      gitlab.com/rarimo/tss/tss-svc.init calls runtime.init
      gitlab.com/rarimo/tss/tss-svc/internal/cli.init calls github.com/prometheus/client_golang/prometheus/promhttp.init, which eventually calls runtime.Version
      gitlab.com/rarimo/tss/tss-svc/internal/config.init calls gitlab.com/distributed_lab/kit/comfig.init, which eventually calls runtime.GOROOT
      gitlab.com/rarimo/tss/tss-svc/internal/tss.init calls github.com/bnb-chain/tss-lib/common.init, which eventually calls runtime.Func.Name
      gitlab.com/rarimo/tss/tss-svc/internal/tss.init calls github.com/bnb-chain/tss-lib/common.init, which eventually calls runtime.FuncForPC
      gitlab.com/rarimo/tss/tss-svc/pkg/types.init calls google.golang.org/protobuf/runtime/protoimpl.init, which eventually calls runtime.KeepAlive
      internal/cli/main.go:116:41: gitlab.com/rarimo/tss/tss-svc/internal/cli.Run calls github.com/bnb-chain/tss-lib/ecdsa/keygen.GeneratePreParams, which eventually calls runtime.NumCPU
      internal/config/cosmos.go:18:20: gitlab.com/rarimo/tss/tss-svc/internal/config.config.Cosmos calls gitlab.com/rarimo/tss/tss-svc/internal/config.Tendermint, which eventually calls runtime.ReadMemStats
      internal/connectors/broadcast.go:64:34: gitlab.com/rarimo/tss/tss-svc/internal/connectors.BroadcastConnector.SubmitToWithReport calls google.golang.org/grpc/status.FromError, which eventually calls runtime.T
ypeAssertionError.Error
      internal/connectors/broadcast.go:64:34: gitlab.com/rarimo/tss/tss-svc/internal/connectors.BroadcastConnector.SubmitToWithReport calls google.golang.org/grpc/status.FromError, which eventually calls runtime.e
rrorString.Error
      internal/connectors/broadcast.go:64:34: gitlab.com/rarimo/tss/tss-svc/internal/connectors.BroadcastConnector.SubmitToWithReport calls google.golang.org/grpc/status.FromError, which eventually calls runtime.p
lainError.Error
      internal/connectors/core.go:81:28: gitlab.com/rarimo/tss/tss-svc/internal/connectors.CoreConnector.SubmitReport calls fmt.Sprint, which eventually calls runtime.Func.FileLine
      internal/core/controllers/util.go:187:46: gitlab.com/rarimo/tss/tss-svc/internal/core/controllers.GetFeeManagementContent calls gitlab.com/rarimo/rarimo-core/x/tokenmanager/types.Network.GetFeeParams, which
eventually calls runtime.GC
      internal/grpc/server_impl.go:48:30: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGRPC calls google.golang.org/grpc.NewServer, which eventually calls runtime.Caller
      internal/grpc/server_impl.go:48:30: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGRPC calls google.golang.org/grpc.NewServer, which eventually calls runtime.Callers
      internal/grpc/server_impl.go:48:30: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGRPC calls google.golang.org/grpc.NewServer, which eventually calls runtime.Stack
      internal/grpc/server_impl.go:50:25: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGRPC calls google.golang.org/grpc.Server.Serve, which eventually calls runtime.Gosched
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.BlockProfile
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.MemProfile
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.MemProfileRecord.InUseObjects
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.MemProfileRecord.Stack
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.MutexProfile
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.ReadTrace
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.SetCPUProfileRate
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.SetMutexProfileFraction
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.StackRecord.Stack
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.StartTrace
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.StopTrace
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls runtime.ThreadCreateProfile
      internal/grpc/server_impl.go:87:29: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.AddOperation calls gitlab.com/distributed_lab/logan/v3.Entry.Error, which eventually calls runtime.CallersFrames
      internal/grpc/server_impl.go:87:29: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.AddOperation calls gitlab.com/distributed_lab/logan/v3.Entry.Error, which eventually calls runtime.Frames.Next
      internal/grpc/server_impl.go:87:29: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.AddOperation calls gitlab.com/distributed_lab/logan/v3.Entry.Error, which eventually calls runtime.NumCgoCall
      internal/grpc/server_impl.go:87:29: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.AddOperation calls gitlab.com/distributed_lab/logan/v3.Entry.Error, which eventually calls runtime.NumGoroutine
      internal/timer/main.go:21:32: gitlab.com/rarimo/tss/tss-svc/internal/timer.NewTimer calls github.com/tendermint/tendermint/rpc/client/http.baseRPCClient.Status, which eventually calls runtime.SetFinalizer
      main.go:16:22: gitlab.com/rarimo/tss/tss-svc.main calls runtime.GOMAXPROCS
      pkg/types/service.pb.go:471:36: gitlab.com/rarimo/tss/tss-svc/pkg/types.MsgAddOperationResponse.String calls google.golang.org/protobuf/internal/impl.Export.MessageStringOf, which eventually calls runtime.Me
mProfileRecord.InUseBytes

Vulnerability #4: GO-2023-1753
  Templates containing actions in unquoted HTML attributes (e.g.
  "attr={{.}}") executed with empty input can result in output
  with unexpected results when parsed due to HTML normalization
  rules. This may allow injection of arbitrary attributes into
  tags.

  More info: https://pkg.go.dev/vuln/GO-2023-1753

  Standard library
```

```
    Found in: html/template@go1.20.3
    Fixed in: html/template@go1.20.4

    Call stacks in your code:
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls html/template.Template.Execute
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls html/template.Template.ExecuteTemplate

Vulnerability #5: GO-2023-1752
  Not all valid JavaScript whitespace characters are considered to
  be whitespace. Templates containing whitespace characters
  outside of the character set "\t\n\f\r\u0020\u2028\u2029" in
  JavaScript contexts that also contain actions may not be
  properly sanitized during execution.

  More info: https://pkg.go.dev/vuln/GO-2023-1752

  Standard library
    Found in: html/template@go1.20.3
    Fixed in: html/template@go1.20.4

    Call stacks in your code:
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls html/template.Template.Execute
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls html/template.Template.ExecuteTemplate

Vulnerability #6: GO-2023-1751
  Angle brackets (<>) are not considered dangerous characters when
  inserted into CSS contexts. Templates containing multiple
  actions separated by a '/' character can result in unexpectedly
  closing the CSS context and allowing for injection of unexpected
  HTML, if executed with untrusted input.

  More info: https://pkg.go.dev/vuln/GO-2023-1751

  Standard library
    Found in: html/template@go1.20.3
    Fixed in: html/template@go1.20.4

    Call stacks in your code:
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls html/template.Template.Execute
      internal/grpc/server_impl.go:69:28: gitlab.com/rarimo/tss/tss-svc/internal/grpc.ServerImpl.RunGateway calls net/http.ListenAndServe, which eventually calls html/template.Template.ExecuteTemplate

=== Informational ===

Found 1 vulnerability in packages that you import, but there are no call
stacks leading to the use of this vulnerability. You may not need to
take any action. See https://pkg.go.dev/golang.org/x/vuln/cmd/govulncheck
for details.

Vulnerability #1: GO-2022-1098
  Erroneous message decoding can cause denial of service. Improper
  checking of maximum witness size during node message decoding
  prevented nodes in Lightning Labs lnd (before 0.15.2-beta) to
  sync.
  More info: https://pkg.go.dev/vuln/GO-2022-1098
  Found in: github.com/btcsuite/btcd@v0.22.0-beta
  Fixed in: github.com/btcsuite/btcd@v0.23.2
```

- Govulncheck spotted several dependencies that should be updated.
- No major issues found by Govulncheck.

5.7 Nancy

Analysis Output Sample:

```
pkg:golang/github.com/btcsuite/btcd@v0.22.0-beta
2 known vulnerabilities affecting installed version

| [CVE-2022-44797] CWE-617: Reachable Assertion

| Description       | btcd before 0.23.2, as used in Lightning Labs lnd before 0.15.2-beta and
|                   | other Bitcoin-related products, mishandles witness size checking.
|                   |
|                   | Sonatype's research suggests that this CVE's details differ from those
|                   | defined at NVD. See
|                   | https://ossindex.sonatype.org/vulnerability/CVE-2022-44797 for details

| OSS Index ID      | CVE-2022-44797

| CVSS Score        | 9.8/10 (Critical)

| CVSS Vector       | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

| Link for more info | https://ossindex.sonatype.org/vulnerability/CVE-2022-44797?component-type=golang&component-name=github.com%2Fbtcsuite%2Fbtcd&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-
dev |

| [CVE-2022-39389] CWE-20: Improper Input Validation

| Description       | Lightning Network Daemon (lnd) is an implementation of a lightning bitcoin
|                   | overlay network node. All lnd nodes before version `v0.15.4` are vulnerable
|                   | to a block parsing bug that can cause a node to enter a degraded state once
|                   | encountered. In this degraded state, nodes can continue to make payments
|                   | and forward HTLCs, and close out channels. Opening channels is prohibited,
|                   | and also on chain transaction events will be undetected. This can cause
|                   | CLTV delta expires forgetting the funds in the HTLC. A patch is available
|                   | in `lnd` version 0.15.4. Users are advised to upgrade. Users unable to
|                   | upgrade may use the `lncli updatechanpolicy` RPC call to increase their
|                   | CLTV value to a very high amount or increase their fee policies. This will
|                   | prevent nodes from routing through your node, meaning that no pending HTLCs
|                   | can be present.
|                   |
|                   | Sonatype's research suggests that this CVE's details differ from those
|                   | defined at NVD. See
|                   | https://ossindex.sonatype.org/vulnerability/CVE-2022-39389 for details

| OSS Index ID      | CVE-2022-39389

| CVSS Score        | 6.5/10 (Medium)

| CVSS Vector       | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:L

| Link for more info | https://ossindex.sonatype.org/vulnerability/CVE-2022-39389?component-type=golang&component-name=github.com%2Fbtcsuite%2Fbtcd&utm_source=nancy-client&utm_medium=integration&utm_content=0.0.0-
dev |

pkg:golang/github.com/ethereum/go-ethereum@v1.10.26
3 known vulnerabilities affecting installed version

| [CVE-2021-42219] CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')

| Description       | Go-Ethereum v1.10.9 was discovered to contain an issue which allows
|                   | attackers to cause a denial of service (DoS) via sending an excessive
|                   | amount of messages to a node. This is caused by missing memory in the
|                   | component /ethash/algorithm.go.
|                   |
|                   | Sonatype's research suggests that this CVE's details differ from those
|                   | defined at NVD. See
```

AUTOMATED TESTING

- Nancy spotted several dependencies that should be updated.
- No major issues found by Nancy.

THANK YOU FOR CHOOSING

// HALBORN