Audit Report

Noble Tariff Module

v1.0

July 8, 2023

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Strangelove Crypto, Inc. to perform a security audit of the Noble Tariff Module.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/strangelove-ventures/noble
Commit	`9e50cfa9ff5208bc9194792fa2c944a5bcf37cd1`
Scope	Only `x/tariff` and its integration into `app/app.go` has been in scope of this audit.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Noble is designed to be the premier asset issuance chain in Cosmos. The `tariff` module runs before the distribution module in the begin block sequence and collects a percentage of the inflation minted every block and distributes it among configured entities.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	Medium	
Code readability and clarity	High	Readability is in line with best practices for Go development and Cosmos SDK-based blockchain applications.
Level of documentation	High	The module included detailed documentation that accurately described its functionality.
Test coverage	Medium	The module is tested by the e2e testing framework interchaintest. It is not possible to compute code coverage metrics with it.

Summary of Findings

No	Description	Severity	Status
1	Attackers can cause a consensus failure by sending coins through IBC	Critical	Resolved
2	Panic in `SendCoinsFromModuleToAccount` causes chain to halt	Critical	Resolved
3	Fee will round to zero for small fees or amounts	Minor	Acknowledged
4	A large number of `DistributionEntities` could slow down and even halt the chain	Informational	Acknowledged
5	Remove unused functions	Informational	Acknowledged
6	Use of magic numbers decreases maintainability	Informational	Acknowledged
7	Inefficient execution of zero funds allocation	Informational	Resolved
8	Redundant check during distribution entity validation	Informational	Acknowledged
9	Inappropriate upper bound of `transferFeeBPS` creates unfavorable conditions for users performing IBC transfers	Informational	Acknowledged
10	Zero shares and duplicate addresses are allowed in distribution entities, which can lead to inefficiencies	Informational	Resolved

Detailed Findings

1. Attackers can cause a consensus failure by sending coins through IBC

Severity: Critical

During the execution of the `BeginBlocker`, a panic occurs in the `AllocateTokens` function in `x/tariff/keeper/allocation.go:30` if the `SendCoinsFromModuleToAccount` function returns an error. This error can be caused if a zero-amount coin is passed to the `SendCoinsFromModuleToAccount` function. Currently, there is no check in the `AllocateTokens` function that prevents a coin with a zero amount from being added to `coins`.

An unsuspecting user or an attacker could cause such a panic by sending a coin with an amount in the `[10_000, 19_999]` range through IBC. Since the scale factor is `10_000` and the result is truncated, this would cause the sending of coins with a zero amount using the `SendCoinsFromModuleToAccount` function, which would cause a panic as described above.

Consequently, the chain will be permanently halted since the error is not recoverable without pruning the state.

A test case able to reproduce this issue can be found in Appendix 1.

Recommendation

We recommend preventing zero-amount fees.

Status: Resolved

2. Panic in `SendCoinsFromModuleToAccount` causes chain to halt

Severity: Critical

When the `BeginBlocker` is executed, there is a possibility of a panic in the `AllocateTokens` function defined in `x/tariff/keeper/allocation.go:30`.

This panic occurs when the `SendCoinsFromModuleToAccount` method of `bankKeeper` encounters an error.

One such error can arise if the recipient's address is listed in the `Bank` module's blacklist. The purpose of blacklisting is to prevent users from sending funds to module accounts or to proactively block specific malicious accounts as determined by the governance.

An attacker could forge a message to send coins to these blacklisted accounts, causing a panic that halts the chain.

Recommendation

We recommend returning an error message instead of panicking if the `SendCoinsFromModule` function returns an error.

Status: Resolved

3. Fee will round to zero for small fees or amounts

Severity: Minor

The `SendPacket` function in `x/tariff/keeper/keeper.go:81-82` introduces a rounding issue that can occur when the `bpsFee` or the `fullAmount` take small values. The rounding will be in favor of the user and will round down to 0.

For example, using the `TransferFeeBps` value of 1 from the docs in `docs/modules/tariff.md`, amounts under `10000uDenom` will have a fee value that rounds down to 0.

Recommendation

We recommend using a ceiling function during fee calculation. This will round in favor of the protocol to ensure that small amounts are also appropriately paying the required transfer fee.

Status: Acknowledged

The client states that this is intended behavior.

4. A large number of `DistributionEntities` could slow down and even halt the chain

Severity: Informational

The `AllocateTokens` function defined in `x/tariff/keeper/allocation.go:7-33` is responsible for distributing fees to `DistributionEntities` during the `BeginBlocker`.

Since it performs an unbounded loop through all the registered `DistributionEntities`, it could cause the `BeginBlocker` execution to take more time than expected.

This could slow down the chain or, in the worst case, halt the node because Tendermint's timeouts are hit.

We classify this issue as informational since only governance can register `DistributionEntities`.

Recommendation

We recommend implementing a hard cap for the number of `DistributionEntities`.

Status: Acknowledged

The client states that they don't foresee a large number of distribution entities at any time in the near future.

5. Remove unused functions

Severity: Informational

There are some instances of unused functions in the codebase:

- The `String` function in `x/tariff/types/params.go:139`.
- The `GetAccount` function in `x/tariff/types/expected_keeper.go:11`.

Recommendation

We recommend removing the unused functions mentioned above.

Status: Acknowledged

6. Use of magic numbers decreases maintainability

Severity: Informational

In the codebase, hard-coded number literals without context or a description are used. Using such "magic numbers" goes against best practices as they reduce code readability and maintenance as developers are unable to easily understand their use and may make inconsistent changes across the codebase.

In `x/tariff/types/params.go:85`, there is a validation check if the value of `transferFeeBPS` is greater than a hard-coded value of `10000`.

In `x/tariff/keeper/keeper.go:81`, the values `4` and `1` are hard-coded.

Recommendation

We recommend defining magic numbers as constants with descriptive variable names and comments, where necessary.

Status: Acknowledged

7. Inefficient execution of zero funds allocation

Severity: Informational

In `x/tariff/keeper/allocation.go:7`, the `AllocateTokens` function continues iteration even if the fees to distribute are zero, which is inefficient.

Recommendation

We recommend short-circuiting the execution if there are no fees to distribute to reduce computation in the `BeginBlocker`.

Status: Resolved

8. Redundant check during distribution entity validation

Severity: Informational

In `x/tariff/types/params.go:55`, the `Share` of a `distributionEntity` is validated to ensure its value is less than `OneDec`. Then, in line `59`, the sum of the `Share` values of all `distributionEntity` is calculated, which is then validated in line `62` to ensure that the sum is less than `OneDec`.

That second validation in line `62` covers the first validation in line `59` which makes the first validation redundant.

Recommendation

We recommend removing the redundant validation in line `59`.

Status: Acknowledged

9. Inappropriate upper bound of `transferFeeBPS` creates unfavorable conditions for users performing IBC transfers

Severity: Informational

In `x/tariff/types/params.go:80`, the `validateTransferFeeBPS` function validates the provided `transferFeeBPS` value. During the validation however, the upper bound for `transferFeeBPS` is `10000`, which is an unfavorable value for users since it implies that the total transfer value gets deducted as the fee.

In the current implementation, `transferFeeMax` provides the inherent upper cap on the fee amount but `transferFeeMax`. However, that value can be set to any number using governance.

We classify this issue as informational since these values can only be set through governance.

Recommendation

We recommend setting lower upper bounds for `transferFeeBPS` and `transferFeeMax`.

Status: Acknowledged

10. Zero shares and duplicate addresses are allowed in distribution entities, which can lead to inefficiencies

Severity: Informational

In `x/tariff/types/params.go:42-60`, the `validateDistributionEntityParams` function allows zero shares for distribution entities and does not filter duplicate addresses. This is inefficient as zero distribution entity shares will receive no fees while duplicate addresses may receive separate fee transfers. Both cases consume computation power unnecessarily.

Recommendation

We recommend rejecting distribution entities with zero shares and enforcing that no duplicate addresses exists in distribution entities.

Status: Resolved

Appendix: Test Cases

1. **Test case for “Attackers can cause a consensus failure by sending coins through IBC”**

The following interchaintest test case reproduces the described issue:

```
package interchaintest_test

import (
	"context"
	"encoding/json"
	"fmt"
	"testing"

	transfertypes "github.com/cosmos/ibc-go/v3/modules/apps/transfer/types"
	"github.com/strangelove-ventures/interchaintest/v3"
	"github.com/strangelove-ventures/interchaintest/v3/chain/cosmos"
	"github.com/strangelove-ventures/interchaintest/v3/ibc"
	"github.com/strangelove-ventures/interchaintest/v3/testreporter"
	"github.com/strangelove-ventures/interchaintest/v3/testutil"
	integration "github.com/strangelove-ventures/noble/interchaintest"
	"github.com/stretchr/testify/require"
	"go.uber.org/zap/zaptest"
)

func TestICS20BPSFees(t *testing.T) {
	if testing.Short() {
		t.Skip()
	}

	t.Parallel()

	ctx := context.Background()

	rep := testreporter.NewNopReporter()
	eRep := rep.RelayerExecReporter(t)

	client, network := interchaintest.DockerSetup(t)

	repo, version := integration.GetDockerImageInfo()

	var (
		noble, gaia          *cosmos.CosmosChain
		roles, roles2        NobleRoles
		extraWallets         ExtraWallets
		paramauthorityWallet Authority
```

```
	)

	chainCfg := ibc.ChainConfig{
		Type:           "cosmos",
		Name:           "noble",
		ChainID:        "noble-1",
		Bin:            "nobled",
		Denom:          "token",
		Bech32Prefix:   "noble",
		CoinType:       "118",
		GasPrices:      "0.0token",
		GasAdjustment:  1.1,
		TrustingPeriod: "504h",
		NoHostMount:    false,
		Images: []ibc.DockerImage{
			{
				Repository: repo,
				Version:    version,
				UidGid:     "1025:1025",
			},
		},
		EncodingConfig: NobleEncoding(),
		PreGenesis: func(cc ibc.ChainConfig) (err error) {
			val := noble.Validators[0]
			err = createTokenfactoryRoles(ctx, &roles,
DenomMetadata_rupee, val, false)
			if err != nil {
				return err
			}
			err = createTokenfactoryRoles(ctx, &roles2,
DenomMetadata_drachma, val, false)
			if err != nil {
				return err
			}
			extraWallets, err = createExtraWalletsAtGenesis(ctx, val)
			if err != nil {
				return err
			}
			paramauthorityWallet, err = createParamAuthAtGenesis(ctx,
val)
			return err
		},
		ModifyGenesis: func(cc ibc.ChainConfig, b []byte) ([]byte, error) {
			g := make(map[string]interface{})
			if err := json.Unmarshal(b, &g); err != nil {
				return nil, fmt.Errorf("failed to unmarshal genesis
file: %w", err)
			}
			if err := modifyGenesisTokenfactory(g, "tokenfactory",
DenomMetadata_rupee, &roles, true); err != nil {
```

```
                        return nil, err
                }
                if err := modifyGenesisTokenfactory(g, "fiat-tokenfactory",
DenomMetadata_drachma, &roles2, false); err != nil {
                        return nil, err
                }
                if err := modifyGenesisParamAuthority(g,
paramauthorityWallet.Authority.Address); err != nil {
                        return nil, err
                }
                if err := modifyGenesisTariffDefaults(g,
paramauthorityWallet.Authority.Address); err != nil {
                        return nil, err
                }
                out, err := json.Marshal(&g)
                if err != nil {
                        return nil, fmt.Errorf("failed to marshal genesis
bytes to json: %w", err)
                }
                return out, nil
        },
    }

    nv := 1
    nf := 0

    cf := interchaintest.NewBuiltinChainFactory(zaptest.NewLogger(t),
[]*interchaintest.ChainSpec{
        {
                ChainConfig:   chainCfg,
                NumValidators: &nv,
                NumFullNodes:  &nf,
        },
        {
                Name:          "gaia",
                Version:       "v9.0.2",
                NumValidators: &nv,
                NumFullNodes:  &nf,
        },
    })

    chains, err := cf.Chains(t.Name())
    require.NoError(t, err)

    r := interchaintest.NewBuiltinRelayerFactory(
        ibc.CosmosRly,
        zaptest.NewLogger(t),
        relayerImage,
    ).Build(t, client, network)
```

```
	noble, gaia = chains[0].(*cosmos.CosmosChain),
chains[1].(*cosmos.CosmosChain)
	path := "p"

	ic := interchaintest.NewInterchain().
		AddChain(noble).
		AddChain(gaia).
		AddRelayer(r, "relayer").
		AddLink(interchaintest.InterchainLink{
			Chain1:  noble,
			Chain2:  gaia,
			Path:    path,
			Relayer: r,
		})

	require.NoError(t, ic.Build(ctx, eRep,
interchaintest.InterchainBuildOptions{
		TestName:  t.Name(),
		Client:    client,
		NetworkID: network,

		SkipPathCreation: false,
	}))
	t.Cleanup(func() {
		_ = ic.Close()
	})

	nobleValidator := noble.Validators[0]

	_, err = nobleValidator.ExecTx(ctx, roles2.MasterMinter.KeyName,
		"fiat-tokenfactory", "configure-minter-controller",
roles2.MinterController.Address, roles2.Minter.Address, "-b", "block",
	)
	require.NoError(t, err, "failed to execute configure minter controller
tx")

	_, err = nobleValidator.ExecTx(ctx, roles2.MinterController.KeyName,
		"fiat-tokenfactory", "configure-minter", roles2.Minter.Address,
"1000000000000"+DenomMetadata_drachma.Base, "-b", "block",
	)
	require.NoError(t, err, "failed to execute configure minter tx")

	_, err = nobleValidator.ExecTx(ctx, roles2.Minter.KeyName,
		"fiat-tokenfactory", "mint", extraWallets.User.Address,
"1000000000000"+DenomMetadata_drachma.Base, "-b", "block",
	)
	require.NoError(t, err, "failed to execute mint to user tx")

	userBalance, err := noble.GetBalance(ctx, extraWallets.User.Address,
DenomMetadata_drachma.Base)
```

```
	require.NoError(t, err, "failed to get user balance")
	require.Equalf(t, int64(1000000000000), userBalance, "failed to mint %s
to user", DenomMetadata_drachma.Base)

	nobleChans, err := r.GetChannels(ctx, eRep, noble.Config().ChainID)
	require.NoError(t, err, "failed to get noble channels")
	require.Len(t, nobleChans, 1, "more than one channel found")
	nobleChan := nobleChans[0]

	gaiaReceiver := "cosmos169xaqmxumqa829gg73nxrenkhhd2mrs36j3vrz"

	err = r.StartRelayer(ctx, eRep, path)
	require.NoError(t, err, "failed to start relayer")
	defer r.StopRelayer(ctx, eRep)

	height, err := noble.Height(ctx)
	require.NoError(t, err, "failed to get noble height")

	prefixedDenom :=
transfertypes.GetPrefixedDenom(nobleChan.Counterparty.PortID,
nobleChan.Counterparty.ChannelID, DenomMetadata_drachma.Base)
	denomTrace := transfertypes.ParseDenomTrace(prefixedDenom)
	ibcDenom := denomTrace.IBCDenom()

	// Every value in the [10000, 19999] range will cause the issue
	const amountToSend = 10000

	tx, err := noble.SendIBCTransfer(ctx, nobleChan.ChannelID,
extraWallets.User.KeyName, ibc.WalletAmount{
		Address: gaiaReceiver,
		Denom:   DenomMetadata_drachma.Base,
		Amount:  amountToSend,
	}, ibc.TransferOptions{})

	require.NoError(t, err, "failed to send ibc transfer from noble")

	_, err = testutil.PollForAck(ctx, noble, height, height+10, tx.Packet)
	require.NoError(t, err, "failed to find ack for ibc transfer")

	userBalance, err = noble.GetBalance(ctx, extraWallets.User.Address,
DenomMetadata_drachma.Base)
	require.NoError(t, err, "failed to get user balance")
	fmt.Println("User balance", userBalance)
	require.Equal(t, int64(1000000000000-amountToSend), userBalance, "user
balance is incorrect")

	receiverBalance, err := gaia.GetBalance(ctx, gaiaReceiver, ibcDenom)
	require.NoError(t, err, "failed to get receiver balance")
	require.Equal(t, int64(amountToSend), receiverBalance, "receiver balance
incorrect")
```

```
}
```

This test case will timeout since the noble chain halted. In fact, by inspecting the noble chain docker container it is reporting:

```
2023-05-03 17:43:50 3:43PM ERR CONSENSUS FAILURE!!! err="0udrachma: invalid
coins" module=consensus stack="goroutine 115
[running]:\nruntime/debug.Stack()\n\t/usr/local/go/src/runtime/debug/stack.go:24
+0x65\ngithub.com/tendermint/tendermint/consensus.(*State).receiveRoutine.func2(
)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:732
+0x4c\npanic({0x1ac3560,
0xc0046f7720})\n\t/usr/local/go/src/runtime/panic.go:884
+0x212\ngithub.com/strangelove-ventures/noble/x/tariff/keeper.Keeper.AllocateTok
ens({{{0x27b1150, 0xc000369060}, 0xc0002064f8, {0x278d230, 0xc000e069c0},
{0x278d280, 0xc000e06a90}, {0xc000215ee0, 0x6, 0x1a}, ...}, ...},
...)\n\t/noble/x/tariff/keeper/allocation.go:30
+0x419\ngithub.com/strangelove-ventures/noble/x/tariff.BeginBlocker(...)\n\t/nob
le/x/tariff/abci.go:11\ngithub.com/strangelove-ventures/noble/x/tariff.AppModule
.BeginBlock(...)\n\t/noble/x/tariff/module.go:148\ngithub.com/cosmos/cosmos-sdk/
types/module.(*Manager).BeginBlock(_, {{0x27a2248, 0xc00005a040}, {0x27b0a28,
0xc003e28500}, {{0xb, 0x0}, {0xc003c67959, 0x7}, 0x27, ...}, ...},
...)\n\t/go/pkg/mod/github.com/cosmos/cosmos-sdk@v0.45.15/types/module/module.go
:491 +0x3bb\ngithub.com/strangelove-ventures/noble/app.(*App).BeginBlocker(_,
{{0x27a2248, 0xc00005a040}, {0x27b0a28, 0xc003e28500}, {{0xb, 0x0},
{0xc003c67959, 0x7}, 0x27, ...}, ...}, ...)\n\t/noble/app/app.go:684
+0x85\ngithub.com/cosmos/cosmos-sdk/baseapp.(*BaseApp).BeginBlock(_,
{{0xc003e177a0, 0x20, 0x20}, {{0xb, 0x0}, {0xc003c67959, 0x7}, 0x27, {0xa9a297e,
...}, ...},
...})\n\t/go/pkg/mod/github.com/cosmos/cosmos-sdk@v0.45.15/baseapp/abci.go:177
+0x97b\ngithub.com/tendermint/tendermint/abci/client.(*localClient).BeginBlockSy
nc(_, {{0xc003e177a0, 0x20, 0x20}, {{0xb, 0x0}, {0xc003c67959, 0x7}, 0x27,
{0xa9a297e, ...}, ...},
...})\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/abci/client/local_cli
ent.go:280
+0x118\ngithub.com/tendermint/tendermint/proxy.(*appConnConsensus).BeginBlockSyn
c(_, {{0xc003e177a0, 0x20, 0x20}, {{0xb, 0x0}, {0xc003c67959, 0x7}, 0x27,
{0xa9a297e, ...}, ...},
...})\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/proxy/app_conn.go:81
+0x55\ngithub.com/tendermint/tendermint/state.execBlockOnProxyApp({0x27a2fd8?,
0xc001149680}, {0x27aab00, 0xc000f7b1c0}, 0xc00269e1e0, {0x27b2660,
0xc0000121b0},
0x26?)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/state/execution.go:3
07 +0x3dd\ngithub.com/tendermint/tendermint/state.(*BlockExecutor).ApplyBlock(_,
{{{0xb, 0x0}, {0xc000e6fd40, 0x7}}, {0xc000e6fd47, 0x7}, 0x1, 0x26,
{{0xc003fe3840, ...}, ...}, ...},
...)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/state/execution.go:140
+0x171\ngithub.com/tendermint/tendermint/consensus.(*State).finalizeCommit(0xc00
0cb5880,
0x27)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:16
```

```
61
+0xafd\ngithub.com/tendermint/tendermint/consensus.(*State).tryFinalizeCommit(0x
c000cb5880,
0x27)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:15
70
+0x2ff\ngithub.com/tendermint/tendermint/consensus.(*State).enterCommit.func1()\
n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:1505
+0xaa\ngithub.com/tendermint/tendermint/consensus.(*State).enterCommit(0xc000cb5
880, 0x27,
0x0)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:154
3
+0xccf\ngithub.com/tendermint/tendermint/consensus.(*State).addVote(0xc000cb5880
, 0xc0038cbae0, {0x0,
0x0})\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:21
65
+0x18dc\ngithub.com/tendermint/tendermint/consensus.(*State).tryAddVote(0xc000cb
5880, 0xc0038cbae0, {0x0?,
0x47c306?})\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state
.go:1963
+0x2c\ngithub.com/tendermint/tendermint/consensus.(*State).handleMsg(0xc000cb588
0, {{0x2787da0?, 0xc004a37a30?}, {0x0?,
0x0?}})\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:
861
+0x170\ngithub.com/tendermint/tendermint/consensus.(*State).receiveRoutine(0xc00
0cb5880,
0x0)\n\t/go/pkg/mod/github.com/cometbft/cometbft@v0.34.27/consensus/state.go:788
+0x505\ncreated by
github.com/tendermint/tendermint/consensus.(*State).OnStart\n\t/go/pkg/mod/githu
b.com/cometbft/cometbft@v0.34.27/consensus/state.go:379 +0x12d\n"
2023-05-03 17:43:50 3:43PM INF service stop impl={"Logger":{}} module=consensus
msg={} wal=/var/cosmos-chain/noble-1/data/cs.wal/wal
2023-05-03 17:43:50 3:43PM INF service stop
impl={"Dir":"/var/cosmos-chain/noble-1/data/cs.wal","Head":{"ID":"0QHgGQB4I0pO:/
var/cosmos-chain/noble-1/data/cs.wal/wal","Path":"/var/cosmos-chain/noble-1/data
/cs.wal/wal"},"ID":"group:0QHgGQB4I0pO:/var/cosmos-chain/noble-1/data/cs.wal/wal
","Logger":{}} module=consensus msg={}
wal=/var/cosmos-chain/noble-1/data/cs.wal/wal
2023-05-03 17:43:52 3:43PM INF Timed out dur=2000 height=39 module=consensus
round=0 step=3
2023-05-03 17:44:01 3:44PM INF Ensure peers module=pex numDialing=0 numInPeers=0
numOutPeers=0 numToDial=10
2023-05-03 17:44:01 3:44PM INF No addresses to dial. Falling back to seeds
module=pex
2023-05-03 17:44:31 3:44PM INF Ensure peers module=pex numDialing=0 numInPeers=0
numOutPeers=0 numToDial=10
2023-05-03 17:44:31 3:44PM INF Saving AddrBook to file
book=/var/cosmos-chain/noble-1/config/addrbook.json module=p2p size=0
2023-05-03 17:44:31 3:44PM INF No addresses to dial. Falling back to seeds
module=pex
```