SHERLOCK

Security Review For Babylon

Public Audit Contest Prepared For: **Babylon**
Lead Security Expert: **dtheo**
Date Audited: **February 4 - March 4, 2025**

Introduction

Phase-2 of the Babylon mainnet will launch a Babylon PoS blockchain as the first Bitcoin Secured Network (BSN). This will be accompanied by the registration of the stakes created during Phase-1 to the Babylon blockchain allowing them to provide finality service for the it and earn PoS rewards for the service.

Executive Summary

The Babylon Chain Launch (Phase-2) audit contest included sign ups of 82 participants. There were three high severity issues and 4 medium severity issues that were found during. The Babylon team addressed each of these issues.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues Found

High	Medium
3	4

Issues Not Fixed and Not Acknowledged

High	Medium
0	0

Security experts who found valid issues

0xNirix
0xeix
LZ_security
n4nika
valuevalk

Issue H-1: Refund mechanism doesn't make sure that there is a fee granter

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/20

Found by

0xeix

Summary

There is currently a refund logic that transfers the funds to a fee payer.

Root Cause

The root cause lies in the fact that the transaction fees can be paid by other enitities and the funds are returned to a fee payer and not a granter.

Internal Pre-conditions

-

External Pre-conditions

Fee granter has to pay for the fees.

Attack Path

Fee grantor paid for the fees but the funds are returned to a fee payer.

Impact

Loss of funds for a fee granter.

PoC

Consider the current refund mechanism:

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/incentive/keeper/refundable_msg_index.go#L10-23

```
// RefundTx refunds the given tx by sending the fee back to the fee payer.
func (k Keeper) RefundTx(ctx context.Context, tx sdk.FeeTx) error {
    txFee := tx.GetFee()
    if txFee.IsZero() {
        // not possible with the global min gas price mechanism
        // but having this check for compatibility in the future
        return nil
    }
    txFeePayer := tx.FeePayer()

    return k.bankKeeper.SendCoinsFromModuleToAccount(ctx, k.feeCollectorName,
    ↪  txFeePayer, txFee)
}
```

The problem is that the fees are refunded to the `feePayer` (the sender of the message) without taking into account the fact that the actual payer can be `feeGranter` resulting in a loss of funds for the granter:

```
type FeeTx interface {
    [Tx](https://pkg.go.dev/github.com/cosmos/cosmos-sdk/types#Tx)
    GetGas() [uint64](https://pkg.go.dev/builtin#uint64)
    GetFee() [Coins](https://pkg.go.dev/github.com/cosmos/cosmos-sdk/types#Coins)
    FeePayer() [][byte](https://pkg.go.dev/builtin#byte)
    FeeGranter() [][byte](https://pkg.go.dev/builtin#byte)
}
```

Mitigation

Check out the Celestia solution for this issue:

https://github.com/rootulp/celestia-app/blob/d10fdbd4e5507f8449747a2388c4657f593b691b/app/posthandler/refund_gas_remaining.go#L133-L138

```
// getRecipient returns the address that should receive the refund.
func getRecipient(feeTx sdk.FeeTx) sdk.AccAddress {
    if feeGranter := feeTx.FeeGranter(); feeGranter != nil {
        return feeGranter
    }
    return feeTx.FeePayer()
}
```

Here the funds are actually sent to the right entity.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/babylonlabs-io/babylon/pull/594

Issue H-2: The EXPIRED judgment does not include the current block

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/33

Found by

LZ_security

Summary

The EXPIRED judgment does not include the current block, fp can still be unbonded after Delegation EXPIRED.

Root Cause

btcHeight+d.UnbondingTime > d.EndHeight should be btcHeight+d.UnbondingTime >= d.EndHeight

Internal Pre-conditions

External Pre-conditions

Attack Path

1. The Delegation of fp is about to expire.
2. fp executes BTCUndelegate in the same block that expires.

Impact

When the unbonded event is executed after the EXPIRED event, a painc occurs and the chain is halted. Or it will result in a double reduction in the`TotalBondedSat` of fp.

PoC

The `BTCUndelegate` function gets the status through `GetStatus`, and Delegation `EXPIRED` returns an error:

```
func (ms msgServer) BTCUndelegate(goCtx context.Context, req
↪  *types.MsgBTCUndelegate) (*types.MsgBTCUndelegateResponse, error) {
    defer telemetry.ModuleMeasureSince(types.ModuleName, time.Now(),
    ↪  types.MetricsKeyBTCUndelegate)
```

```
    ctx := sdk.UnwrapSDKContext(goCtx)
    // basic stateless checks
    if err := req.ValidateBasic(); err != nil {
        return nil, status.Errorf(codes.InvalidArgument, "%v", err)
    }

    btcDel, bsParams, err := ms.getBTCDelWithParams(ctx, req.StakingTxHash)

    if err != nil {
        return nil, err
    }

    // ensure the BTC delegation with the given staking tx hash is active
    btcTip := ms.btclcKeeper.GetTipInfo(ctx)

->  btcDelStatus := btcDel.GetStatus(
        btcTip.Height,
        bsParams.CovenantQuorum,
    )

->  if btcDelStatus == types.BTCDelegationStatus_UNBONDED || btcDelStatus ==
 ↪  types.BTCDelegationStatus_EXPIRED {
        return nil, types.ErrInvalidBTCUndelegateReq.Wrap("cannot unbond an unbonded
         ↪  BTC delegation")
    }
    ......
}
```

The problem is that `GetStatus` does not include the current block in its judgment. If the current block is in `d.EndHeight - d.UnbondingTime`, `BTCDelegationStatus_EXPIRED` will not be returned:

```
func (d *BTCDelegation) GetStatus(
    btcHeight uint32,
    covenantQuorum uint32,
) BTCDelegationStatus {
    ......

->   // btcHeight = currentBtcTipHeight
->   // if btcHeight > d.EndHeight - d.UnbondingTime
->  if btcHeight+d.UnbondingTime > d.EndHeight {
        return BTCDelegationStatus_EXPIRED
    }

    return BTCDelegationStatus_ACTIVE
}
```

The EXPIRED event executes at: `d.EndHeight - d.UnbondingTime`

```
func (k Keeper) AddBTCDelegation(
    ctx sdk.Context,
    btcDel *types.BTCDelegation,
) error {
    ......
    // record event that the BTC delegation will become expired (unbonded) at
    ↪  EndHeight-w
    // This event will be generated to subscribers as block event, when the
    // btc light client block height will reach btcDel.EndHeight-wValue
    expiredEvent := types.NewEventPowerDistUpdateWithBTCDel(&types.EventBTCDelegati
    ↪  onStateUpdate{
        StakingTxHash: stakingTxHash.String(),
        NewState:      types.BTCDelegationStatus_EXPIRED,
    })

    // NOTE: we should have verified that EndHeight > btcTip.Height + unbonding_time
->  k.addPowerDistUpdateEvent(ctx, btcDel.EndHeight-btcDel.UnbondingTime,
↪   expiredEvent)
    ......
}
```

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/btcstaking/types/btc_delegation.go#L145-L147

So `BTCUndelegate` and `EXPIRED` can be executed in the same block.

When handling EXPIRED events, delegater's unbond state is determined first(IsUnbondedEarly), but since the event is handled in `BeginBlocker` and sent before `BTCUndelegate`, so `DelegatorUnbondingInfo` is not set:

```
func (k Keeper) ProcessAllPowerDistUpdateEvents(
    ctx context.Context,
    dc *ftypes.VotingPowerDistCache,
    events []*types.EventPowerDistUpdate,
) *ftypes.VotingPowerDistCache {
            ........
            case types.BTCDelegationStatus_UNBONDED:
                // add the unbonded BTC delegation to the map
                //@audit-info unbonded fp    fpByBtcPkHex      UNBONDED   EXPIRED   ?
                k.processPowerDistUpdateEventUnbond(ctx, fpByBtcPkHex, btcDel,
                ↪  unbondedSatsByFpBtcPk)
            case types.BTCDelegationStatus_EXPIRED:
                types.EmitExpiredDelegationEvent(sdkCtx, delStkTxHash)

->              // IsUnbondedEarly -> return
↪   d.BtcUndelegation.DelegatorUnbondingInfo != nil
->              if !btcDel.IsUnbondedEarly() {
                    // only adds to the new unbonded list if it hasn't
```

```
                    // previously unbonded with types.BTCDelegationStatus_UNBONDED
                    k.processPowerDistUpdateEventUnbond(ctx, fpByBtcPkHex, btcDel,
                    ↪  unbondedSatsByFpBtcPk)
                }
            }
```

This leads to a painc:

UpdatePowerDist -> ProcessAllPowerDistUpdateEvents -> MustProcessBtcDelegationUnbonded :

```
func (k Keeper) MustProcessBtcDelegationUnbonded(ctx context.Context, fp, del
↪  sdk.AccAddress, sats uint64) {
    err := k.IncentiveKeeper.BtcDelegationUnbonded(ctx, fp, del, sats)
->  if err != nil {
->      panic(err)
    }
}

func (k Keeper) BtcDelegationUnbonded(ctx context.Context, fp, del sdk.AccAddress,
↪  sat uint64) error {
    amtSat := sdkmath.NewIntFromUint64(sat)
    return k.btcDelegationModifiedWithPreInitDel(ctx, fp, del, func(ctx
    ↪  context.Context, fp, del sdk.AccAddress) error {
->      return k.subDelegationSat(ctx, fp, del, amtSat)
    })
}


func (k Keeper) subDelegationSat(ctx context.Context, fp, del sdk.AccAddress, amt
↪  sdkmath.Int) error {
    btcDelRwdTracker, err := k.GetBTCDelegationRewardsTracker(ctx, fp, del)
    if err != nil {
        return err
    }

->  btcDelRwdTracker.SubTotalActiveSat(amt)
->  if btcDelRwdTracker.TotalActiveSat.IsNegative() {
->      return types.ErrBTCDelegationRewardsTrackerNegativeAmount
    }
    if err := k.setBTCDelegationRewardsTracker(ctx, fp, del, btcDelRwdTracker); err
    ↪  != nil {
        return err
    }

    return k.subFinalityProviderStaked(ctx, fp, amt)
}
```

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/finality/keeper/power_dist_change.go#L437-L442

The event is handled in `BeginBlocker`, and painc causes the chain to halt.

finality/abci.go/BeginBlocker -> UpdatePowerDist -> ProcessAllPowerDistUpdateEvents

The`amt` passed by the`subDelegationSat` function, which is actually the amount of delegation, is set when`AddBTCDelegation` is added.

`UpdatePowerDist` function, through `k.BTCStakingKeeper.GetBTCDelegation(ctx, delStkTxHash)` get this value and use this value to update `fp.TotalBondedSat`. If UNBOND is repeated, `fp.TotalBondedSat` will also be reduced repeatedly.

Mitigation

```
func (d *BTCDelegation) GetStatus(
    btcHeight uint32,
    covenantQuorum uint32,
) BTCDelegationStatus {

    ......
-   if btcHeight+d.UnbondingTime > d.EndHeight {
+   if btcHeight+d.UnbondingTime >= d.EndHeight {
        return BTCDelegationStatus_EXPIRED
    }

    return BTCDelegationStatus_ACTIVE
}
```

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/babylonlabs-io/babylon/pull/597

Issue H-3: If the Covenant signature does not pass, EXPIRED events it will still be executed.

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/35

Found by

LZ_security

Summary

If the Covenant signature does not pass , EXPIRED events it will still be executed, causes `fp.TotalBondedSat` to be incorrectly reduced.

Root Cause

If the Covenant signature does not pass , EXPIRED events it will still be executed.

Internal Pre-conditions

External Pre-conditions

Attack Path

1. Delegater uses the `CreateBTCDelegation` command to create a Delegation. The `EXPIRED` message is sent.
2. The Covenants began to sign.
3. Delegater unstake before the number of signatures reaches the threshold.
4. The signature fails. The Active message cannot be sent, but the EXPIRED message is sent.
5. The EXPIRED file is executed after a period of time.

Impact

`fp.TotalBondedSat` to be incorrectly reduced.

PoC

Create a Delegation, if by `AddBTCDelegationInclusionProof` function, `activeEvent` and `expiredEvent` will be added at the same time:

```
func (ms msgServer) AddBTCDelegationInclusionProof(
    goCtx context.Context,
    req *types.MsgAddBTCDelegationInclusionProof,
) (*types.MsgAddBTCDelegationInclusionProofResponse, error) {
    ......
    activeEvent := types.NewEventPowerDistUpdateWithBTCDel(
        &types.EventBTCDelegationStateUpdate{
            StakingTxHash: stakingTxHash.String(),
            NewState:      types.BTCDelegationStatus_ACTIVE,
        },
    )

    ms.addPowerDistUpdateEvent(ctx, timeInfo.TipHeight, activeEvent)

    // record event that the BTC delegation will become unbonded at EndHeight-w
    expiredEvent := types.NewEventPowerDistUpdateWithBTCDel(&types.EventBTCDelegati
    ↪  onStateUpdate{
        StakingTxHash: req.StakingTxHash,
        NewState:      types.BTCDelegationStatus_EXPIRED,
    })

    // NOTE: we should have verified that EndHeight > btcTip.Height +
    ↪  min_unbonding_time
    ms.addPowerDistUpdateEvent(ctx, btcDel.EndHeight-params.UnbondingTimeBlocks,
    ↪  expiredEvent)
......
}
```

The problem is that if the `CreateBTCDelegation` function is used, the `expiredEvent` is sent first, and the `activeEvent` is sent after the Covenants signature is passed. If the signature does not pass the activeEvent, it will never be sent, but `expiredEvent` has been sent and will eventually execute:

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/btcstaking/keeper/msg_server.go#L294-L297

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/btcstaking/keeper/btc_delegations.go#L76-L83

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/btcstaking/keeper/btc_delegations.go#L134-L141

If staker/Delegater is unstake in the btc chain after the `CreateBTCDelegation` enters the Covenants signature phase,Covenants should not sign:

In addition,`BTCUndelegate` causes the signature to fail:

```
func (ms msgServer) AddCovenantSigs(goCtx context.Context, req
↪  *types.MsgAddCovenantSigs) (*types.MsgAddCovenantSigsResponse, error) {
```

```
    ......
    // ensure BTC delegation is still pending, i.e., not unbonded
    btcTipHeight := ms.btclcKeeper.GetTipInfo(ctx).Height
    status := btcDel.GetStatus(btcTipHeight, params.CovenantQuorum)
->  if status == types.BTCDelegationStatus_UNBONDED || status ==
↪   types.BTCDelegationStatus_EXPIRED {
        ms.Logger(ctx).Debug("Received covenant signature after the BTC delegation
        ↪  is already unbonded", "covenant pk", req.Pk.MarshalHex())
        return nil, types.ErrInvalidCovenantSig.Wrap("the BTC delegation is already
        ↪  unbonded")
    }

    ......
}
```

Covenants If the signature is not passed, the `BTCUndelegate` can be executed only when the `UNBONDED` or `EXPIRED` status is unavailable Undelegate, So staker can block Covenants' signatures by using `BTCUndelegate`.

```
func (ms msgServer) BTCUndelegate(goCtx context.Context, req
↪   *types.MsgBTCUndelegate) (*types.MsgBTCUndelegateResponse, error) {
    defer telemetry.ModuleMeasureSince(types.ModuleName, time.Now(),
    ↪  types.MetricsKeyBTCUndelegate)

    ctx := sdk.UnwrapSDKContext(goCtx)
    // basic stateless checks
    if err := req.ValidateBasic(); err != nil {
        return nil, status.Errorf(codes.InvalidArgument, "%v", err)
    }

    btcDel, bsParams, err := ms.getBTCDelWithParams(ctx, req.StakingTxHash)

    if err != nil {
        return nil, err
    }

    // ensure the BTC delegation with the given staking tx hash is active
    btcTip := ms.btclcKeeper.GetTipInfo(ctx)

->  btcDelStatus := btcDel.GetStatus(
        btcTip.Height,
        bsParams.CovenantQuorum,
    )

->  if btcDelStatus == types.BTCDelegationStatus_UNBONDED || btcDelStatus ==
↪   types.BTCDelegationStatus_EXPIRED {
        return nil, types.ErrInvalidBTCUndelegateReq.Wrap("cannot unbond an unbonded
        ↪  BTC delegation")
    }
    ......
```

```
}
```

expiredEvent causes fp's `TotalBondedSat` to decrease, or a negative number in the system causes painc.

The balance of Delegater is set upon creation and stored in the Store:

```
// CreateBTCDelegation creates a BTC delegation
func (ms msgServer) CreateBTCDelegation(goCtx context.Context, req
 ↪  *types.MsgCreateBTCDelegation) (*types.MsgCreateBTCDelegationResponse, error) {
    ......

    // 7.all good, construct BTCDelegation and insert BTC delegation
    // NOTE: the BTC delegation does not have voting power yet. It will
    // have voting power only when it receives a covenant signatures
    newBTCDel := &types.BTCDelegation{
        StakerAddr:       parsedMsg.StakerAddress.String(),
        BtcPk:            parsedMsg.StakerPK.BIP340PubKey,
        Pop:              parsedMsg.ParsedPop,
        FpBtcPkList:      parsedMsg.FinalityProviderKeys.PublicKeysBbnFormat,
        StakingTime:      uint32(parsedMsg.StakingTime),
        StartHeight:      timeInfo.StartHeight,
        EndHeight:        timeInfo.EndHeight,
->      TotalSat:         uint64(parsedMsg.StakingValue),
        StakingTx:        parsedMsg.StakingTx.TransactionBytes,
        StakingOutputIdx: paramsValidationResult.StakingOutputIdx,
        SlashingTx:       types.NewBtcSlashingTxFromBytes(parsedMsg.StakingSlashing
         ↪  Tx.TransactionBytes),
        DelegatorSig:     parsedMsg.StakerStakingSlashingTxSig.BIP340Signature,
        UnbondingTime:    uint32(parsedMsg.UnbondingTime),
        CovenantSigs:     nil, // NOTE: covenant signature will be submitted in a
         ↪  separate msg by covenant
        BtcUndelegation: &types.BTCUndelegation{
            UnbondingTx:              parsedMsg.UnbondingTx.TransactionBytes,
            SlashingTx:               types.NewBtcSlashingTxFromBytes(parsedMsg.Unb
             ↪  ondingSlashingTx.TransactionBytes),
            DelegatorSlashingSig:
             ↪  parsedMsg.StakerUnbondingSlashingSig.BIP340Signature,
            CovenantSlashingSigs:     nil, // NOTE: covenant signature will be
             ↪  submitted in a separate msg by covenant
            CovenantUnbondingSigList: nil, // NOTE: covenant signature will be
             ↪  submitted in a separate msg by covenant
            DelegatorUnbondingInfo:   nil,
        },
        ParamsVersion: paramsVersion,      // version of the params against which
         ↪  delegation was validated
        BtcTipHeight:  timeInfo.TipHeight, // height of the BTC light client tip at
         ↪  the time of the delegation creation
    }
```

```
    // add this BTC delegation, and emit corresponding events
->  if err := ms.AddBTCDelegation(ctx, newBTCDel); err != nil {
        panic(fmt.Errorf("failed to add BTC delegation that has passed verification:
        ↪  %w", err))
    }
```

When handling `expiredEvent` events, the balance of Delegater is read and fp TotalBondedSat is reduced:

```
func (k Keeper) ProcessAllPowerDistUpdateEvents(
    ctx context.Context,
    dc *ftypes.VotingPowerDistCache,
    events []*types.EventPowerDistUpdate,
) *ftypes.VotingPowerDistCache {
    // a map where key is finality provider's BTC PK hex and value is a list
    // of BTC delegations satoshis amount that newly become active under this
    ↪  provider
    activedSatsByFpBtcPk := map[string][]uint64{}
    // a map where key is finality provider's BTC PK hex and value is a list
    // of BTC delegations satoshis that were unbonded or expired without previously
    // being unbonded
    unbondedSatsByFpBtcPk := map[string][]uint64{}
    // a map where key is slashed finality providers' BTC PK
    slashedFPs := map[string]struct{}{}
    // a map where key is jailed finality providers' BTC PK
    jailedFPs := map[string]struct{}{}
    // a map where key is unjailed finality providers' BTC PK
    unjailedFPs := map[string]struct{}{}

    // simple cache to load fp by his btc pk hex
    fpByBtcPkHex := map[string]*types.FinalityProvider{}

    /*
        filter and classify all events into new/expired BTC delegations and
        ↪  jailed/slashed FPs
    */
    sdkCtx := sdk.UnwrapSDKContext(ctx)
    for _, event := range events {
        switch typedEvent := event.Ev.(type) {
        case *types.EventPowerDistUpdate_BtcDelStateUpdate:
            delEvent := typedEvent.BtcDelStateUpdate
            delStkTxHash := delEvent.StakingTxHash

->          btcDel, err := k.BTCStakingKeeper.GetBTCDelegation(ctx, delStkTxHash)
            if err != nil {
                panic(err) // only programming error
            }

            switch delEvent.NewState {
            case types.BTCDelegationStatus_ACTIVE:
```

```
                // newly active BTC delegation
                // add the BTC delegation to each restaked finality provider
                for _, fpBTCPK := range btcDel.FpBtcPkList {
                    fpBTCPKHex := fpBTCPK.MarshalHex()
                    activedSatsByFpBtcPk[fpBTCPKHex] =
                     ↪  append(activedSatsByFpBtcPk[fpBTCPKHex], btcDel.TotalSat)
                }

                k.processRewardTracker(ctx, fpByBtcPkHex, btcDel, func(fp, del
                 ↪  sdk.AccAddress, sats uint64) {
                    k.MustProcessBtcDelegationActivated(ctx, fp, del, sats)//
                })
            case types.BTCDelegationStatus_UNBONDED:
                // add the unbonded BTC delegation to the map
->              k.processPowerDistUpdateEventUnbond(ctx, fpByBtcPkHex, btcDel,
 ↪  unbondedSatsByFpBtcPk)
            case types.BTCDelegationStatus_EXPIRED:
                types.EmitExpiredDelegationEvent(sdkCtx, delStkTxHash)

                // IsUnbondedEarly -> return
                 ↪  d.BtcUndelegation.DelegatorUnbondingInfo != nil
                if !btcDel.IsUnbondedEarly() {
                    // only adds to the new unbonded list if it hasn't
                    // previously unbonded with types.BTCDelegationStatus_UNBONDED
->                  k.processPowerDistUpdateEventUnbond(ctx, fpByBtcPkHex, btcDel,
 ↪  unbondedSatsByFpBtcPk)
                }
            }
            .....


            // process all new unbonding BTC delegations under this finality
             ↪  provider
            if fpUnbondedSats, ok := unbondedSatsByFpBtcPk[fpBTCPKHex]; ok {
                // handle unbonded delegations for this finality provider
                for _, unbodedSats := range fpUnbondedSats {
->                    // RemoveBondedSats -> v.TotalBondedSat -= sats
->                  fp.RemoveBondedSats(unbodedSats)
                }
                // remove the finality provider entry in fpUnbondedSats map, so that
                // after the for loop the rest entries in fpUnbondedSats belongs to
                 ↪  new
                // finality providers that might have btc delegations entries
                // that activated and unbonded in the same slice of events
                delete(unbondedSatsByFpBtcPk, fpBTCPKHex)
            }
            ......
        }
    }
```

```
func (k Keeper) processPowerDistUpdateEventUnbond(
    ctx context.Context,
    cacheFpByBtcPkHex map[string]*types.FinalityProvider,
    btcDel *types.BTCDelegation,
    unbondedSatsByFpBtcPk map[string][]uint64,
) {
    for _, fpBTCPK := range btcDel.FpBtcPkList {
        fpBTCPKHex := fpBTCPK.MarshalHex()
->      unbondedSatsByFpBtcPk[fpBTCPKHex] =
 ↪  append(unbondedSatsByFpBtcPk[fpBTCPKHex], btcDel.TotalSat)
    }
    k.processRewardTracker(ctx, cacheFpByBtcPkHex, btcDel, func(fp, del
     ↪  sdk.AccAddress, sats uint64) {
        k.MustProcessBtcDelegationUnbonded(ctx, fp, del, sats)
    })
}
```

Mitigation

Send EXPIRED and Active messages at the same time instead of separately.

Executive Summary

The issue leads to:

- either unfair reduction of the staker voting power
- or potential panic if the staker did not have any other delegation and had 0 voting power

The error is triggered if quorum of the covenant committee members did not send their covenant signatures before the delegation had expired.

Protocol team fixed the issue in following PR: #634.

This PR changes the logic of the voting power updates so that expired events for delegations without covenant quorum are ignored.

Issue M-1: Btcstaking module allows `stakingTx` to be coinbase transaction which is unslashable for 100 blocks

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/6

Found by

n4nika

Summary

Coinbase transactions have a special property of not being spendable for 100 bocks after creation. If now a staker uses such a transaction as a staking transaction (by adding the required outputs), that transaction will be recognized as a valid staking TX but if the owner of it double-signs, he cannot be slashed for 100 blocks due to the unspendability of the coinbase TX.

Root Cause

Looking at CreateBTCDelegation and ValidateParsedMessageAgainstTheParams which does verification on the provided TX, there are no specific checks for whether a transaction is a coinbase transaction.

Internal Pre-conditions

None

External Pre-conditions

Attacker needs to be the creator of a block (a miner) in order to build the coinbase TX like they want

Attack Path

- Create a coinbase TX which is a valid staking TX
- Call `CreateBTCDelegation` with that TX
- It gets accepted and its value added as voting power

Impact

The README states under `High` impact: `Inability to slash BTC stake for which the voting power was involved in double-signing.`

That is exactly what happens here. Even if the staker's delegator misbehaves, the staker's `stakingTx` cannot be spent until 100 blocks after. Adding to the impact, if now the minimum staking time is less than 100 bitcoin blocks, this allows the malicious staker to unstake before getting slashed (if the slashing even gets retried after 100 blocks)

PoC

No response

Mitigation

Coinbase transactions can be identified since the `ID` of their input must be all zeros. Therefore consider checking whether a staking transaction has one input with an ID of all-zeros and reject it if so.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/babylonlabs-io/babylon/pull/563

Issue M-2: Message is indexed as refundable even if the signature was over a fork

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/18

Found by

0xeix

Summary

When adding a finality sig, a message is still marked as refundable even if the vote was over the fork.

Root Cause

The root cause lies in the fact that there is no return after slashing the finality provider resulting in a message being indexed as refundable in the `IncentiveKeeper`.

Internal Pre-conditions

-

External Pre-conditions

A FP adds a finality sig.

Attack Path

A FP adds a finality sig and his message is incorrectly marked as refundable due to the missing return.

Impact

The message can still be refunded even if it's a signature addition with invalid data.

PoC

The current code makes no return when slashing a provider that adds a finality sig:

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/finality/keeper/msg_server.go#L192-216

```
// if this finality provider has signed the canonical block before,
    // slash it via extracting its secret key, and emit an event
    if ms.HasEvidence(ctx, req.FpBtcPk, req.BlockHeight) {
        // the finality provider has voted for a fork before!
        // If this evidence is at the same height as this signature, slash this
        ↪  finality provider

        // get evidence
        evidence, err := ms.GetEvidence(ctx, req.FpBtcPk, req.BlockHeight)
        if err != nil {
            panic(fmt.Errorf("failed to get evidence despite HasEvidence returns
            ↪  true"))
        }

        // set canonical sig to this evidence
        evidence.CanonicalFinalitySig = req.FinalitySig
        ms.SetEvidence(ctx, evidence)

        // slash this finality provider, including setting its voting power to
        // zero, extracting its BTC SK, and emit an event
        ms.slashFinalityProvider(ctx, req.FpBtcPk, evidence)
    }

    // at this point, the finality signature is 1) valid, 2) over a canonical block,
    // and 3) not duplicated.
    // Thus, we can safely consider this message as refundable
    ms.IncentiveKeeper.IndexRefundableMsg(ctx, req)
```

As you can see here, after slashing a provider, the message is marked as refundable:

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/incentive/keeper/refundable_msg_index.go#L23-38

```
// IndexRefundableMsg indexes the given refundable message by its hash.
func (k Keeper) IndexRefundableMsg(ctx context.Context, msg sdk.Msg) {
    msgHash := types.HashMsg(msg)
    err := k.RefundableMsgKeySet.Set(ctx, msgHash)
    if err != nil {
        panic(err) // encoding issue; this can only be a programming error
    }
}
```

The problem is that there has to be some conditions to be satisfied outlined in the comments for the message to be refundable:

```
// at this point, the finality signature is 1) valid, 2) over a canonical block,
    // and 3) not duplicated.
    // Thus, we can safely consider this message as refundable
```

The problem is that there is no return after slashing the provider previously meaning the next line executed will be the indexing of the message. In contrary, there is another slashing logic in the same function that correctly returns the response:

https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/finality/keeper/msg_server.go#L172-178

```
// save evidence
    ms.SetEvidence(ctx, evidence)

    // NOTE: we should NOT return error here, otherwise the state change triggered
    ↪  in this tx
    // (including the evidence) will be rolled back
    return &types.MsgAddFinalitySigResponse{}, nil
}
```

Mitigation

After slashing a finality provider, add a response with nil as an error so the message is not marked as refundable later.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/babylonlabs-io/babylon/pull/592

Issue M-3: maybeResendFromStore may wrongly submit a checkpoint transaction twice

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/23

Found by

valuevalk

Summary

`maybeResendFromStore` in `relayer.go` in `vigilante module`, currently resends the same checkpoint tx again, no matter what the error returned from the json-rpc function `GetRawTransactionFunc` is, which in some situations could obstruct the correct flow, as we could end up submitting the same transaction twice.

Root Cause

Bitcoin rpc errors: https://github.com/btcsuite/btcd/blob/bb52d7d78d9cf335e0611b9ae06ad8c77e75de0b/btcjson/jsonrpcerr.go#L177

When submitting a checkpoint tx to bitcoin, we use `maybeResendFromStore` which has a fail-safe mechanism which only proceeds with sending the transaction if there is an error when calling `GetRawTransactionFunc`, this is to prevent sending already send transactions, if for example the submitter service is restarted.

However there is a flaw, currently we do that no matter what the returned error is, this poses a problem as there is possibility that error could be returned due to problems with the rpc(rate limits exceeded, network error & etc), or other json-rpc error codes that are not `transaction not found`.

```
func (rl *Relayer) SendCheckpointToBTC(ckpt
↪  *ckpttypes.RawCheckpointWithMetaResponse) error {
.........
    if rl.shouldSendCompleteCkpt(ckptEpoch) || rl.shouldSendTx2(ckptEpoch) {
@>>         hasBeenProcessed, err := maybeResendFromStore(
            ckptEpoch,
            rl.store.LatestCheckpoint,
            rl.GetRawTransaction,
            rl.sendTxToBTC,
        )
        if err != nil {
            return err
        }
        if hasBeenProcessed {
            return nil
```

```
        }
    }
....
}

@>> // maybeResendFromStore - checks if we need to resubmit txns from a store
@>> // in case "submitter" service was restarted, we want to ensure that we don't
↪  send txns again for a checkpoint
@>> // that has already been processed.
@>> // Returns true if the first transactions are in the mempool (no resubmission
↪  needed),
@>> // and false if any transaction was re-sent from the store.
func maybeResendFromStore(
    epoch uint64,
    getLatestStoreCheckpoint GetLatestCheckpointFunc,
    getRawTransaction GetRawTransactionFunc,
    sendTransaction SendTransactionFunc,
) (bool, error) {
    storedCkpt, exists, err := getLatestStoreCheckpoint()
    if err != nil {
        return false, err
    } else if !exists {
        return false, nil
    }
    if storedCkpt.Epoch != epoch {
        return false, nil
    }
    maybeResendFunc := func(tx *wire.MsgTx) error {
        txID := tx.TxHash()
@>> _, err = getRawTransaction(&txID) // todo(lazar): check for specific not found
↪  err
@>> if err != nil {
@>>     _, err := sendTransaction(tx)
            if err != nil {
@>>         return err //we could end up here
            }
            // we know about this tx, but we needed to resend it from already
            ↪  constructed tx from db
            return nil
        }
        // tx exists in mempool and is known to us
        return nil
    }
@>> if err := maybeResendFunc(storedCkpt.Tx1); err != nil {
        return false, err
    }
@>> if err := maybeResendFunc(storedCkpt.Tx2); err != nil {
        return false, err
    }
    return true, nil
```

```
}
```

A few bad scenarios could happen here:

- If in reality `storedCkpt.Tx2` needs to be resend, but we failed on `maybeResendFunc(storedCkpt.Tx1)` because we falsely send the same tx1 again, we would not be able to re-submit `storedCkpt.Tx2`.
- The `MaybeResubmitSecondCheckpointTx` flow will not be executed because we call `continue`.

```
func (s *Submitter) processCheckpoints() {
    defer s.wg.Done()
    quit := s.quitChan()

    for {
        select {
        case ckpt := <-s.poller.GetSealedCheckpointChan():
            s.logger.Infof("A sealed raw checkpoint for epoch %v is found",
            ↪  ckpt.Ckpt.EpochNum)
            if err := s.relayer.SendCheckpointToBTC(ckpt); err != nil {
                s.logger.Errorf("Failed to submit the raw checkpoint for %v: %v",
                ↪  ckpt.Ckpt.EpochNum, err)
                s.metrics.FailedCheckpointsCounter.Inc()

@>>             continue
            }
            if err := s.relayer.MaybeResubmitSecondCheckpointTx(ckpt); err != nil {
                s.logger.Errorf("Failed to resubmit the raw checkpoint for %v: %v",
                ↪  ckpt.Ckpt.EpochNum, err)
                s.metrics.FailedCheckpointsCounter.Inc()
            }
            s.metrics.SecondsSinceLastCheckpointGauge.Set(0)
        case <-quit:
            // We have been asked to stop
            return
        }
    }
}
```

Attack Path:

1. getRawTransaction returns an error, due to network connection (for example)
2. sendTransaction(tx) is called, however it fails with error as its already submitted to the bitcoin chain
3. Error is propagated from `relayer.go maybeResendFromStore` to `submitter.go`
4. As result even though checkpoint 2 needed to be re-submitted its not. (Either with

the `maybeResendFunc(storedCkpt.Tx2)` flow, or with the bump fee flow in MaybeResubmitSecondCheckpointTx)

Impact

If sendTx is called for the 1st checkpoint tx, when the tx is already on bitcoin chain, the operation will fail and will obstruct the flow, possibly leading to not correctly submitting the 2nd tx if it needed to be resubmitted.

Recommendation

The correct behaviours would be to handle separately RPC/curl request error or any json-rpc error code different from ”transaction not found”.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/babylonlabs-io/vigilante/pull/250

Issue M-4: Incorrect BTC Delegation Reward Calculation Due to Using Current Stake Amount Instead of Historical Stake

Source: https://github.com/sherlock-audit/2024-12-babylon-judging/issues/70

Found by

0xNirix

Summary

Missing historical stake tracking in the incentive reward system will cause incorrect reward distribution for BTC delegators as changing delegation amounts after a period but before rewards are calculated will result in rewards based on current stake rather than historical stake.

Root Cause

In https://github.com/sherlock-audit/2024-12-babylon/blob/main/babylon/x/incentive/keeper/reward_tracker.go#L200 the `calculateDelegationRewardsBetween` function calculates rewards using the delegation's current active satoshi amount rather than the historical amount that was active during the reward period. This means:

```
// This incorrectly uses current stake amount (btcDelRwdTracker.TotalActiveSat)
// rather than the historical amount that was active during that period
rewardsWithDecimals :=
 ↪  differenceWithDecimals.MulInt(btcDelRwdTracker.TotalActiveSat)
```

This issue exists because the `BTCDelegationRewardsTracker` struct stores only the current stake amount and starting period, without preserving historical stake amounts per period. When delegations are modified (through unbonding or new activations), the stake amount is immediately updated, but there's no record of what it was at different periods. However, rewards of a block may be delayed (they are always delayed by FinalitySigTimeout to allow for voting time, but can be massively delayed when governance resumes finality after halt). Finality providers are rewarded in proportion to the correct cached total bonded delegations, however individual delegators may receive incorrect reward out of that based on their current stake.

Internal Pre-conditions

NA

External Pre-conditions

NA

Attack Path

1. **Delegator A** has 100,000 sats delegated to a finality provider during Period 1
2. Rewards accrue but are not yet calculated and distributed as finality has halted.
3. Before rewards are calculated, **Delegator A** reduces their delegation to 50,000 sats (period changes to Period 2)
4. A governance action triggers calculation of rewards for Period 1 after resuming finality
5. The system incorrectly uses the current stake amount (50,000 sats) to calculate Period 1's rewards instead of the original amount (100,000 sats)
6. **Delegator A** receives only half the rewards they should have earned

Alternatively:

1. **Delegator B** has 50,000 sats delegated to a finality provider during Period 1
2. Rewards accrue for Period 1 but are not yet calculated and distributed as finality has halted.
3. Before rewards are calculated, **Delegator B** increases their delegation to 100,000 sats (period changes to Period 2)
4. A governance action triggers calculation of rewards for Period 1
5. The system incorrectly uses the current stake amount (100,000 sats) to calculate Period 1's rewards
6. **Delegator B** receives double the rewards they should have earned

Please not this issue can happen (even without finality halt and governance resumption) for `FinalitySigTimeout` blocks, as protocol waits for `FinalitySigTimeout` blocks before finalizing a block's reward and unbonding/ bonding in that period can cause this issue.

Impact

The BTC delegators suffer from incorrect reward distribution. Those who decrease their stake before rewards are calculated receive fewer rewards than they deserve, while those who increase their stake receive more rewards than they deserve. This creates unfairness in the protocol and could potentially be exploited by sophisticated users who understand the timing of reward calculations specially in cases of governance resumption of finality.

PoC

To demonstrate the issue with concrete examples, let's walk through how the incentive system handles reward calculations when delegations change, and why this leads to incorrect reward distribution.

Scenario: Delegator Reduces Stake After Period 1 But Before Rewards Are Calculated Due to Governance Action

Initial Setup

1. `DelegatorA` has 100,000 sats staked to `FinalityProviderX` during Period 1
2. A finality halt occurs during Period 1, preventing rewards from being distributed
3. Period 2 begins with the halt still in effect if delegator unbonds

Step 1: Delegation Reduction in Period 2

```
// DelegatorA calls to unbond 50,000 sats creating Period 2
k.BtcDelegationUnbonded(ctx, fpAddr, delegatorAddr, 50000)

// This function calls:
k.btcDelegationModifiedWithPreInitDel(ctx, fp, del, function(ctx, fp, del) {
    return k.subDelegationSat(ctx, fp, del, 50000)
})

// Which increments the period and calculates rewards for current period:
endedPeriod, err := k.IncrementFinalityProviderPeriod(ctx, fp)

// The delegation's TotalActiveSat is reduced:
if err := k.subDelegationSat(ctx, fp, del, amtSat); err != nil {
    return err
}
```

Here, the system:

1. Increments the period from 1 to 2
2. Attempts to calculate rewards for Period 1, but since there was a finality halt, no rewards are distributed yet
3. Reduces the delegation's `TotalActiveSat` from 100,000 to 50,000
4. Initializes a new `BTCDelegationRewardsTracker` with 50,000 sats and a starting period of 2

Step 2: Governance Action to Resume Finality Later, a governance proposal is submitted to resume finality:

```
// HandleResumeFinalityProposal is called with the finality providers to jail
k.HandleResumeFinalityProposal(ctx, fpPksHex, haltingHeight)

// This jails the specified FPs and recalculates voting power for previous heights
for h := uint64(haltingHeight); h <= uint64(currentHeight); h++ {
    distCache := k.GetVotingPowerDistCache(ctx, h)
    // ... updating distCache ...
    k.SetVotingPowerDistCache(ctx, h, distCache)
}

// Then it tallies blocks
k.TallyBlocks(ctx)
```

Step 3: Reward Distribution The `TallyBlocks` finalizes and eventually leads to reward distribution for previously finalized blocks in `HandleRewarding` (called via EndBlocker)

```
// HandleRewarding calls the reward to stakers if the block is finalized
func (k Keeper) HandleRewarding(ctx context.Context, targetHeight int64) {
    ...
    for height := nextHeightToReward; height <= uint64(targetHeight); height++ {

        if !block.Finalized {
            continue
        }
        k.rewardBTCStaking(ctx, height)

    }

}
```

Step 4: Reward Calculation in `RewardBTCStaking` Now, `RewardBTCStaking` but after DelegatorA's stake has been reduced:

```
// RewardBTCStaking distributes rewards based on voting power
func (k Keeper) RewardBTCStaking(ctx context.Context, height uint64, dc
↪  *ftypes.VotingPowerDistCache, voters map[string]struct{}) {
    // Get the gauge for this height
    gauge := k.GetBTCStakingGauge(ctx, height)

    // Calculate total voting power of voters
    var totalVotingPowerOfVoters uint64
    for i, fp := range dc.FinalityProviders {
        if _, ok := voters[fp.BtcPk.MarshalHex()]; ok {
            totalVotingPowerOfVoters += fp.TotalBondedSat
        }
    }
```

```
    // Distribute rewards to FPs and delegations
    for _, fp := range dc.FinalityProviders {
        if _, ok := voters[fp.BtcPk.MarshalHex()]; !ok {
            continue
        }

        // Calculate portion of rewards for this FP
        fpPortion := sdkmath.LegacyNewDec(int64(fp.TotalBondedSat)). => this is
        ↪  cached total bonded delegations to the F
            QuoTruncate(sdkmath.LegacyNewDec(int64(totalVotingPowerOfVoters)))
        coinsForFpsAndDels := gauge.GetCoinsPortion(fpPortion)

        // Give commission to FP
        coinsForCommission := types.GetCoinsPortion(coinsForFpsAndDels,
        ↪  *fp.Commission)
        k.accumulateRewardGauge(ctx, types.FinalityProviderType, fp.GetAddress(),
        ↪  coinsForCommission)

        // Distribute remaining rewards to delegations
        coinsForBTCDels := coinsForFpsAndDels.Sub(coinsForCommission...)
        k.AddFinalityProviderRewardsForBtcDelegations(ctx, fp.GetAddress(),
        ↪  coinsForBTCDels)
    }
}
```

Important Note: The finality provider's share of rewards is correctly calculated based on the cached voting power distribution (`dc`) from. This is properly preserved in the system and is not affected by the issue. The problem only occurs at the individual delegator level when rewards are calculated for each delegator.

This leads to:

```
// AddFinalityProviderRewardsForBtcDelegations adds rewards to the FP's current
↪  rewards
func (k Keeper) AddFinalityProviderRewardsForBtcDelegations(ctx context.Context, fp
↪  sdk.AccAddress, rwd sdk.Coins) error {
    fpCurrentRwd, err := k.GetFinalityProviderCurrentRewards(ctx, fp)
    if err != nil {
        return err
    }

    // Add rewards to the FP's current rewards pool
    fpCurrentRwd.AddRewards(rwd)
    return k.setFinalityProviderCurrentRewards(ctx, fp, fpCurrentRwd)
}
```

Step 5: Later, When Delegator Withdraws Rewards When `DelegatorA` later calls to withdraw rewards:

```
// MsgWithdrawReward handler calls:
```

```
k.sendAllBtcDelegationTypeToRewardsGauge(ctx, sType, addr)

// Which calls:
k.sendAllBtcRewardsToGauge(ctx, del)

// Which iterates through all FPs associated with this delegator:
k.iterBtcDelegationsByDelegator(ctx, del, func(del, fp sdk.AccAddress) error {
    return k.btcDelegationModified(ctx, fp, del)
})

// This calls:
k.btcDelegationModifiedWithPreInitDel(ctx, fp, del, func(ctx context.Context, fp,
↪  del sdk.AccAddress) error { return nil })

// Which leads to:
endedPeriod, err := k.IncrementFinalityProviderPeriod(ctx, fp)

// And then:
k.CalculateBTCDelegationRewardsAndSendToGauge(ctx, fp, del, endedPeriod)

// Which calls:
rewards, err := k.CalculateBTCDelegationRewards(ctx, fp, del, endPeriod)

// Finally leading to the problematic calculation:
func (k Keeper) calculateDelegationRewardsBetween(
    ctx context.Context,
    fp sdk.AccAddress,
    btcDelRwdTracker types.BTCDelegationRewardsTracker,
    endingPeriod uint64,
) (sdk.Coins, error) {
    // ... get historical rewards ...

    // Calculate difference in rewards per satoshi
    differenceWithDecimals :=
    ↪  ending.CumulativeRewardsPerSat.Sub(starting.CumulativeRewardsPerSat...)

    // HERE IS THE ISSUE:
    // It uses current stake (50,000) instead of correct historical stake (100,000)
    rewardsWithDecimals :=
    ↪  differenceWithDecimals.MulInt(btcDelRwdTracker.TotalActiveSat)

    rewards := rewardsWithDecimals.QuoInt(types.DecimalAccumulatedRewards)
    return rewards, nil
}
```

Result: `DelegatorA` receives only 5 BBN in rewards instead of the 10 BBN they should have received based on their stake during Period 1, because the rewards were calculated using their current stake (50,000 sats) rather than their historical stake (100,000 sats).

Even worse, in a similar way if `DelegatorA` increases their stake, it will unfairly steal

rewards of other delegators.

Mitigation

Due to the below, we have decided to postpone the resolution of this issue and have it as part of a planned refactoring of our rewards distribution implementation.

Executive Summary

This issue might lead to minor discrepancies in the case of delayed rewards distribution, the Babylon team finds that the impact of this is minimal, especially under circumstances of substantial economic security and therefore wide distribution of rewards:

- the finality_sig_timeout parameter is set to 3, therefore in most cases delay won't impact stakers rewards in noticable way as blocks will be BTC finalized right after comet-bft blocks
- rewards are correctly distributed between the finality providers
- even big changes in stakers voting power impact their rewards in small way

Scope

Repository: babylonlabs-io/babylon

Audited Commit: 64a86ebbc46184b2b36554fc23a0f349482f4f87

Final Commit: c00be14f35217d4100338aee2b283d84a60904b8

Files:

- app/ante/ante.go
- app/ante/ante_btc_validation_decorator.go
- app/ante/fee_checker.go
- app/app.go
- app/encoding.go
- app/export.go
- app/genesis.go
- app/include_upgrade_mainnet.go
- app/include_upgrade_testnet.go
- app/keepers/keepers.go
- app/keepers/keys.go
- app/keepers/utils.go
- app/modules.go
- app/params/config.go
- app/params/encoding.go
- app/params/proto.go
- app/upgrades/types.go
- app/upgrades/v1/mainnet/btc_headers.go
- app/upgrades/v1/mainnet/btcstaking_params.go
- app/upgrades/v1/mainnet/cosmwasm_params.go
- app/upgrades/v1/mainnet/finality_params.go
- app/upgrades/v1/mainnet/token_distribution.go
- app/upgrades/v1/testnet/btc_headers.go
- app/upgrades/v1/testnet/btcstaking_params.go
- app/upgrades/v1/testnet/cosmwasm_params.go
- app/upgrades/v1/testnet/finality_params.go

- app/upgrades/v1/testnet/token_distribution.go
- app/upgrades/v1/types.go
- app/upgrades/v1/upgrades.go
- btcstaking/errors.go
- btcstaking/identifiable_staking.go
- btcstaking/identifiable_staking_types.go
- btcstaking/scripts_utils.go
- btcstaking/staking.go
- btcstaking/types.go
- btcstaking/witness_utils.go
- btctxformatter/formatter.go
- client/client/client.go
- client/client/keys.go
- client/client/log.go
- client/client/retry_utils.go
- client/client/tx.go
- client/config/babylon_config.go
- client/config/babylon_query_config.go
- client/query/btccheckpoint.go
- client/query/btclightclient.go
- client/query/btcstaking.go
- client/query/checkpointing.go
- client/query/client.go
- client/query/epoching.go
- client/query/finality.go
- client/query/incentive.go
- client/query/monitor.go
- client/query/staking.go
- client/query/tendermint.go
- cmd/babylond/cmd/create_bls_key.go
- cmd/babylond/cmd/custom_babylon_config.go

- cmd/babylond/cmd/flags.go
- cmd/babylond/cmd/genaccounts.go
- cmd/babylond/cmd/genesis.go
- cmd/babylond/cmd/genhelpers/cmd.go
- cmd/babylond/cmd/genhelpers/set_btc_delegations.go
- cmd/babylond/cmd/genhelpers/set_btc_headers.go
- cmd/babylond/cmd/genhelpers/set_finality_providers.go
- cmd/babylond/cmd/root.go
- cmd/babylond/cmd/sizes_cmd.go
- cmd/babylond/cmd/testnet.go
- cmd/babylond/cmd/validate_genesis.go
- cmd/babylond/main.go
- crypto/bip322/bip322.go
- crypto/bip322/witness.go
- crypto/bls12381/bls.go
- crypto/bls12381/doc.go
- crypto/bls12381/types.go
- crypto/ecdsa/ecdsa.go
- crypto/eots/eots.go
- crypto/eots/error.go
- crypto/schnorr-adaptor-signature/keys.go
- crypto/schnorr-adaptor-signature/sig.go
- crypto/schnorr-adaptor-signature/sign_utils.go
- types/btc_config.go
- types/btc_header_bytes.go
- types/btc_header_hash_bytes.go
- types/btc_schnorr_eots.go
- types/btc_schnorr_pk.go
- types/btc_schnorr_pub_rand.go
- types/btc_schnorr_sig.go
- types/btcutils.go

- types/errors.go
- types/signer_config.go
- types/utils.go
- wasmbinding/bindings/query.go
- wasmbinding/bindings/utils.go
- wasmbinding/wasm.go
- x/btccheckpoint/abci.go
- x/btccheckpoint/client/cli/query.go
- x/btccheckpoint/client/cli/query_params.go
- x/btccheckpoint/client/cli/tx.go
- x/btccheckpoint/genesis.go
- x/btccheckpoint/keeper/grpc_query.go
- x/btccheckpoint/keeper/grpc_query_params.go
- x/btccheckpoint/keeper/hooks.go
- x/btccheckpoint/keeper/keeper.go
- x/btccheckpoint/keeper/msg_server.go
- x/btccheckpoint/keeper/params.go
- x/btccheckpoint/keeper/submissions.go
- x/btccheckpoint/module.go
- x/btccheckpoint/types/btcutils.go
- x/btccheckpoint/types/codec.go
- x/btccheckpoint/types/errors.go
- x/btccheckpoint/types/expected_keepers.go
- x/btccheckpoint/types/genesis.go
- x/btccheckpoint/types/incentive.go
- x/btccheckpoint/types/keys.go
- x/btccheckpoint/types/msgs.go
- x/btccheckpoint/types/params.go
- x/btccheckpoint/types/query.go
- x/btccheckpoint/types/types.go
- x/btclightclient/client/cli/query.go

- x/btclightclient/client/cli/tx.go
- x/btclightclient/genesis.go
- x/btclightclient/keeper/base_btc_header.go
- x/btclightclient/keeper/grpc_query.go
- x/btclightclient/keeper/hooks.go
- x/btclightclient/keeper/keeper.go
- x/btclightclient/keeper/msg_server.go
- x/btclightclient/keeper/params.go
- x/btclightclient/keeper/state.go
- x/btclightclient/keeper/triggers.go
- x/btclightclient/keeper/utils.go
- x/btclightclient/module.go
- x/btclightclient/types/btc_header_info.go
- x/btclightclient/types/btc_light_client.go
- x/btclightclient/types/codec.go
- x/btclightclient/types/errors.go
- x/btclightclient/types/expected_keepers.go
- x/btclightclient/types/genesis.go
- x/btclightclient/types/hooks.go
- x/btclightclient/types/keys.go
- x/btclightclient/types/msgs.go
- x/btclightclient/types/params.go
- x/btclightclient/types/querier.go
- x/btclightclient/types/query.go
- x/btclightclient/types/types.go
- x/btclightclient/types/utils.go
- x/btclightclient/types/work.go
- x/btcstaking/abci.go
- x/btcstaking/client/cli/query.go
- x/btcstaking/client/cli/query_params.go
- x/btcstaking/client/cli/tx.go

- x/btcstaking/client/cli/utils.go
- x/btcstaking/genesis.go
- x/btcstaking/keeper/btc_delegations.go
- x/btcstaking/keeper/btc_delegators.go
- x/btcstaking/keeper/btc_height_index.go
- x/btcstaking/keeper/finality_providers.go
- x/btcstaking/keeper/genesis.go
- x/btcstaking/keeper/grpc_query.go
- x/btcstaking/keeper/inclusion_proof.go
- x/btcstaking/keeper/keeper.go
- x/btcstaking/keeper/msg_server.go
- x/btcstaking/keeper/params.go
- x/btcstaking/keeper/power_dist_change.go
- x/btcstaking/keeper/query.go
- x/btcstaking/keeper/query_params.go
- x/btcstaking/module.go
- x/btcstaking/types/btc_delegation.go
- x/btcstaking/types/btc_slashing_tx.go
- x/btcstaking/types/btc_undelegation.go
- x/btcstaking/types/btcstaking.go
- x/btcstaking/types/codec.go
- x/btcstaking/types/create_delegation_parser.go
- x/btcstaking/types/errors.go
- x/btcstaking/types/events.go
- x/btcstaking/types/expected_keepers.go
- x/btcstaking/types/genesis.go
- x/btcstaking/types/inclusion_proof.go
- x/btcstaking/types/keys.go
- x/btcstaking/types/metrics.go
- x/btcstaking/types/mocked_keepers.go
- x/btcstaking/types/msg.go

- x/btcstaking/types/params.go
- x/btcstaking/types/pop.go
- x/btcstaking/types/query.go
- x/btcstaking/types/validate_parsed_message.go
- x/checkpointing/abci.go
- x/checkpointing/client/cli/query.go
- x/checkpointing/client/cli/tx.go
- x/checkpointing/client/cli/utils.go
- x/checkpointing/genesis.go
- x/checkpointing/keeper/bls_signer.go
- x/checkpointing/keeper/ckpt_state.go
- x/checkpointing/keeper/genesis_bls.go
- x/checkpointing/keeper/grpc_query_bls.go
- x/checkpointing/keeper/grpc_query_checkpoint.go
- x/checkpointing/keeper/hooks.go
- x/checkpointing/keeper/keeper.go
- x/checkpointing/keeper/msg_server.go
- x/checkpointing/keeper/registration_state.go
- x/checkpointing/keeper/val_bls_set.go
- x/checkpointing/module.go
- x/checkpointing/proposal.go
- x/checkpointing/proposal_expected_keeper.go
- x/checkpointing/types/codec.go
- x/checkpointing/types/errors.go
- x/checkpointing/types/events.go
- x/checkpointing/types/expected_keepers.go
- x/checkpointing/types/genesis.go
- x/checkpointing/types/hooks.go
- x/checkpointing/types/keys.go
- x/checkpointing/types/msgs.go
- x/checkpointing/types/pop.go

- x/checkpointing/types/querier.go
- x/checkpointing/types/query.go
- x/checkpointing/types/types.go
- x/checkpointing/types/utils.go
- x/checkpointing/types/val_bls_set.go
- x/checkpointing/vote_ext.go
- x/epoching/abci.go
- x/epoching/client/cli/query.go
- x/epoching/client/cli/tx.go
- x/epoching/genesis.go
- x/epoching/keeper/epoch_msg_queue.go
- x/epoching/keeper/epoch_slashed_val_set.go
- x/epoching/keeper/epoch_val_set.go
- x/epoching/keeper/epochs.go
- x/epoching/keeper/grpc_query.go
- x/epoching/keeper/hooks.go
- x/epoching/keeper/keeper.go
- x/epoching/keeper/lifecycle_delegation.go
- x/epoching/keeper/lifecycle_validator.go
- x/epoching/keeper/modified_staking.go
- x/epoching/keeper/msg_server.go
- x/epoching/keeper/params.go
- x/epoching/keeper/staking_functions.go
- x/epoching/module.go
- x/epoching/types/codec.go
- x/epoching/types/epoching.go
- x/epoching/types/errors.go
- x/epoching/types/events.go
- x/epoching/types/expected_keepers.go
- x/epoching/types/genesis.go
- x/epoching/types/hooks.go

- x/epoching/types/keys.go
- x/epoching/types/msg.go
- x/epoching/types/params.go
- x/epoching/types/query.go
- x/epoching/types/types.go
- x/epoching/types/validator.go
- x/finality/abci.go
- x/finality/client/cli/query.go
- x/finality/client/cli/query_params.go
- x/finality/client/cli/tx.go
- x/finality/genesis.go
- x/finality/keeper/evidence.go
- x/finality/keeper/genesis.go
- x/finality/keeper/grpc_query.go
- x/finality/keeper/indexed_blocks.go
- x/finality/keeper/liveness.go
- x/finality/keeper/msg_server.go
- x/finality/keeper/params.go
- x/finality/keeper/power_dist_change.go
- x/finality/keeper/power_table.go
- x/finality/keeper/public_randomness.go
- x/finality/keeper/query.go
- x/finality/keeper/query_params.go
- x/finality/keeper/signing_info.go
- x/finality/keeper/tallying.go
- x/finality/keeper/votes.go
- x/finality/module.go
- x/finality/types/codec.go
- x/finality/types/errors.go
- x/finality/types/events.go
- x/finality/types/expected_keepers.go

- x/finality/types/finality.go
- x/finality/types/genesis.go
- x/finality/types/keys.go
- x/finality/types/metrics.go
- x/finality/types/mocked_keepers.go
- x/finality/types/msg.go
- x/finality/types/params.go
- x/finality/types/power_table.go
- x/finality/types/query.go
- x/finality/types/signing_info.go
- x/finality/types/types.go
- x/incentive/abci.go
- x/incentive/client/cli/query.go
- x/incentive/client/cli/query_params.go
- x/incentive/client/cli/tx.go
- x/incentive/genesis.go
- x/incentive/keeper/btc_staking_gauge.go
- x/incentive/keeper/grpc_query.go
- x/incentive/keeper/intercept_fee_collector.go
- x/incentive/keeper/keeper.go
- x/incentive/keeper/msg_server.go
- x/incentive/keeper/params.go
- x/incentive/keeper/query_params.go
- x/incentive/keeper/refund_tx_decorator.go
- x/incentive/keeper/refundable_msg_index.go
- x/incentive/keeper/reward_gauge.go
- x/incentive/module.go
- x/incentive/types/codec.go
- x/incentive/types/errors.go
- x/incentive/types/expected_keepers.go
- x/incentive/types/genesis.go

- x/incentive/types/incentive.go
- x/incentive/types/keys.go
- x/incentive/types/msg.go
- x/incentive/types/params.go
- x/incentive/types/types.go
- x/mint/abci.go
- x/mint/client/cli/query.go
- x/mint/keeper/genesis.go
- x/mint/keeper/grpc_query.go
- x/mint/keeper/keeper.go
- x/mint/module.go
- x/mint/simulation/decoder.go
- x/mint/types/constants.go
- x/mint/types/events.go
- x/mint/types/expected_keepers.go
- x/mint/types/genesis.go
- x/mint/types/keys.go
- x/mint/types/minter.go
- x/monitor/client/cli/query.go
- x/monitor/client/cli/tx.go
- x/monitor/genesis.go
- x/monitor/keeper/grpc_query.go
- x/monitor/keeper/hooks.go
- x/monitor/keeper/keeper.go
- x/monitor/module.go
- x/monitor/types/errors.go
- x/monitor/types/expected_keepers.go
- x/monitor/types/genesis.go
- x/monitor/types/keys.go

Repository: babylonlabs-io/babylon-proto-ts

Audited Commit: 5f30f285c26361fe7de867158a5f2ebdc57d0549

Final Commit: 5f30f285c26361fe7de867158a5f2ebdc57d0549

Files:

- package.json
- scripts/build-proto.js
- src/index.ts

Repository: babylonlabs-io/babylon-staking-indexer

Audited Commit: 12d22e16a3a19a879167b85025deb895b468d8dd

Final Commit: db9071bedaf3f87e7396a27ecf6a8ac41f38bd0e

Files:

- cmd/babylon-staking-indexer/cli/root.go
- cmd/babylon-staking-indexer/main.go
- consumer/event_consumer.go
- internal/clients/base/base_client.go
- internal/clients/bbnclient/bbnclient.go
- internal/clients/bbnclient/interface.go
- internal/clients/bbnclient/types.go
- internal/clients/btcclient/btcclient.go
- internal/clients/btcclient/interface.go
- internal/clients/btcclient/notifier.go
- internal/config/bbn.go
- internal/config/btc.go
- internal/config/config.go
- internal/config/db.go
- internal/config/metrics.go
- internal/config/poller.go
- internal/db/dbclient.go
- internal/db/delegation.go
- internal/db/error.go

- internal/db/finality_provider.go
- internal/db/interface.go
- internal/db/last_processed_height.go
- internal/db/model/delegation.go
- internal/db/model/finality_provider.go
- internal/db/model/last_processed_height.go
- internal/db/model/params.go
- internal/db/model/setup.go
- internal/db/model/timelock.go
- internal/db/params.go
- internal/db/timelock.go
- internal/observability/metrics/metrics.go
- internal/observability/tracing/tracing.go
- internal/services/bootstrap.go
- internal/services/consumer_events.go
- internal/services/delegation.go
- internal/services/delegation_helpers.go
- internal/services/events.go
- internal/services/expiry_checker.go
- internal/services/finality_provider.go
- internal/services/global_params.go
- internal/services/service.go
- internal/services/subscription.go
- internal/services/watch_btc_events.go
- internal/types/error.go
- internal/types/state.go
- internal/utils/poller/poller.go
- internal/utils/rand.go
- internal/utils/rand_test.go
- internal/utils/strconv.go
- internal/utils/utils.go

Repository: babylonlabs-io/bbn-core-ui

Audited Commit: edb3d7a5ffaf3c8dc522968b6a5ddd8d4c64f6e4

Final Commit: 9a6a5548d36049884d32d48fc7cb55c88e2152f9

Files:

- package-lock.json
- package.json
- src/components/Accordion/Accordion.tsx
- src/components/Accordion/components/AccordionDetails.tsx
- src/components/Accordion/components/AccordionSummary.tsx
- src/components/Accordion/hooks/useHeightObserver.ts
- src/components/Accordion/index.ts
- src/components/Avatar/Avatar.tsx
- src/components/Avatar/AvatarGroup.tsx
- src/components/Avatar/index.ts
- src/components/Button/Button.tsx
- src/components/Button/IconButton.tsx
- src/components/Button/index.tsx
- src/components/Card/Card.tsx
- src/components/Card/index.tsx
- src/components/Chip/Chip.tsx
- src/components/Chip/index.ts
- src/components/Dialog/Dialog.tsx
- src/components/Dialog/MobileDialog.tsx
- src/components/Dialog/components/Backdrop.tsx
- src/components/Dialog/components/DialogBody.tsx
- src/components/Dialog/components/DialogFooter.tsx
- src/components/Dialog/components/DialogHeader.tsx
- src/components/Dialog/index.ts
- src/components/Form/Checkbox.tsx
- src/components/Form/FormControl.tsx

- src/components/Form/Input.tsx
- src/components/Form/Radio.tsx
- src/components/Form/Select.tsx
- src/components/Form/components/Toggle.tsx
- src/components/Form/index.tsx
- src/components/Heading/Heading.tsx
- src/components/Heading/index.ts
- src/components/Loader/Loader.tsx
- src/components/Loader/index.ts
- src/components/Popover/Popover.tsx
- src/components/Popover/index.ts
- src/components/Portal/Portal.tsx
- src/components/Portal/index.ts
- src/components/Table/Table.tsx
- src/components/Table/components/Cell.tsx
- src/components/Table/components/Column.tsx
- src/components/Table/components/Row.tsx
- src/components/Table/index.ts
- src/components/Table/types/index.ts
- src/components/Text/Text.tsx
- src/components/Text/index.ts
- src/context/Dialog.context.tsx
- src/context/Table.context.tsx
- src/hooks/useClickOutside.ts
- src/hooks/useControlledState.ts
- src/hooks/useMemoizedArray.ts
- src/hooks/useModalManager.ts
- src/hooks/useResizeObserver.ts
- src/hooks/useTableScroll.ts
- src/hooks/useTableSort.ts
- src/index.tsx

- src/utils/css.ts
- src/vite-env.d.ts
- src/widgets/form/CheckboxField/CheckboxField.tsx
- src/widgets/form/CheckboxField/index.ts
- src/widgets/form/Form/Form.tsx
- src/widgets/form/Form/index.tsx
- src/widgets/form/HiddenField/HiddenField.tsx
- src/widgets/form/HiddenField/index.ts
- src/widgets/form/NumberField/NumberField.tsx
- src/widgets/form/NumberField/index.ts
- src/widgets/form/RadioField/RadioField.tsx
- src/widgets/form/RadioField/index.ts
- src/widgets/form/SelectField/SelectField.tsx
- src/widgets/form/SelectField/index.ts
- src/widgets/form/TextField/TextField.tsx
- src/widgets/form/TextField/index.ts
- src/widgets/form/hooks.ts
- src/widgets/form/types.ts

Repository: babylonlabs-io/bbn-wallet-connect

Audited Commit: 93eeeaf08febbf2fab745117d072c1772507aa3f

Final Commit: 4e6f99b5e1180e07bd55529907c408fdc5999bc5

Files:

- package.json
- src/components/ChainButton/index.tsx
- src/components/Chains/container.tsx
- src/components/Chains/index.tsx
- src/components/ConnectedWallet/index.tsx
- src/components/FieldControl/index.tsx
- src/components/Inscriptions/container.tsx
- src/components/Inscriptions/index.tsx

- src/components/Loader/index.tsx
- src/components/TermsOfService/container.tsx
- src/components/TermsOfService/index.tsx
- src/components/WalletButton/index.tsx
- src/components/WalletProvider/components/Screen.tsx
- src/components/WalletProvider/components/WalletDialog.tsx
- src/components/WalletProvider/constants.tsx
- src/components/WalletProvider/index.tsx
- src/components/Wallets/container.tsx
- src/components/Wallets/index.tsx
- src/context/Chain.context.tsx
- src/context/Inscriptions.context.tsx
- src/context/State.context.tsx
- src/core/Wallet.ts
- src/core/WalletConnector.ts
- src/core/index.ts
- src/core/types.ts
- src/core/utils/bip322.ts
- src/core/utils/mempool.ts
- src/core/utils/wallet.ts
- src/core/wallets/bbn/index.ts
- src/core/wallets/bbn/injectable/index.ts
- src/core/wallets/bbn/keplr/index.ts
- src/core/wallets/bbn/keplr/provider.ts
- src/core/wallets/bbn/leap/index.ts
- src/core/wallets/bbn/leap/provider.ts
- src/core/wallets/bbn/okx/index.ts
- src/core/wallets/bbn/okx/provider.ts
- src/core/wallets/btc/index.ts
- src/core/wallets/btc/injectable/index.ts
- src/core/wallets/btc/keystone/index.ts

- src/core/wallets/btc/keystone/provider.ts
- src/core/wallets/btc/okx/index.ts
- src/core/wallets/btc/okx/provider.ts
- src/core/wallets/btc/onekey/index.ts
- src/core/wallets/btc/onekey/provider.ts
- src/core/wallets/btc/unisat/index.ts
- src/core/wallets/btc/unisat/provider.ts
- src/core/wallets/index.tsx
- src/hooks/useChainConnector.ts
- src/hooks/usePersistState.ts
- src/hooks/useWalletConnect.ts
- src/hooks/useWalletConnectors.tsx
- src/hooks/useWalletWidgets.tsx
- src/hooks/useWidgetState.ts
- src/index.tsx
- src/utils/wallet.ts

Repository: babylonlabs-io/bindings

Audited Commit: 282d81d92eebaf2f150183ac13ebe65c9fc01787

Final Commit: 282d81d92eebaf2f150183ac13ebe65c9fc01787

Files:

- packages/bindings/src/lib.rs
- packages/bindings/src/querier.rs
- packages/bindings/src/query.rs
- packages/bindings/src/types.rs

Repository: babylonlabs-io/btc-staker

Audited Commit: 931bc4eb2c6f9ad82f8e1463c8e319dde189ebd4

Final Commit: 7f44227e5c8dbb15455e6ad0d518c85fcb16a3f8

Files:

- babylonclient/babyloncontroller.go

- babylonclient/interface.go
- babylonclient/msgsender.go
- babylonclient/pop.go
- babylonclient/utils.go
- cmd/stakercli/daemon/daemoncommands.go
- cmd/stakercli/helpers/flags.go
- cmd/stakercli/helpers/json.go
- cmd/stakercli/main.go
- cmd/stakercli/transaction/parsers.go
- cmd/stakercli/transaction/transactions.go
- cmd/stakerd/main.go
- metrics/prometheus.go
- metrics/staker.go
- staker/babylontypes.go
- staker/commands.go
- staker/events.go
- staker/feeestimator.go
- staker/nodebackend.go
- staker/responses.go
- staker/stakerapp.go
- staker/stakercontroller.go
- staker/staking.go
- staker/types.go
- stakercfg/babylon.go
- stakercfg/bitcoind.go
- stakercfg/btcd.go
- stakercfg/config.go
- stakercfg/dbcfg.go
- stakercfg/log.go
- stakercfg/metrics.go
- stakercfg/utils.go

- stakerdb/errors.go
- stakerdb/paginator.go
- stakerdb/trackedtranactionstore.go
- stakerservice/client/rpcclient.go
- stakerservice/service.go
- stakerservice/stakerdresponses.go
- types/feeestimation.go
- types/nodebackend.go
- types/walletbackend.go
- utils/utils.go
- walletcontroller/client.go
- walletcontroller/interface.go
- walletcontroller/transaction.go

Repository: babylonlabs-io/btc-staking-ts

Audited Commit: e2f1004a227d61db705509fe8a17efd7c2971fda

Final Commit: e2f1004a227d61db705509fe8a17efd7c2971fda

Files:

- src/constants/dustSat.ts
- src/constants/fee.ts
- src/constants/internalPubkey.ts
- src/constants/keys.ts
- src/constants/psbt.ts
- src/constants/registry.ts
- src/constants/transaction.ts
- src/constants/unbonding.ts
- src/error/index.ts
- src/index.ts
- src/staking/index.ts
- src/staking/manager.ts
- src/staking/observable/index.ts

- src/staking/observable/observableStakingScript.ts
- src/staking/psbt.ts
- src/staking/stakingScript.ts
- src/staking/transactions.ts
- src/types/UTXO.ts
- src/types/covenantSignatures.ts
- src/types/index.ts
- src/types/params.ts
- src/types/psbtOutputs.ts
- src/types/transaction.ts
- src/utils/babylon.ts
- src/utils/btc.ts
- src/utils/fee/index.ts
- src/utils/fee/utils.ts
- src/utils/index.ts
- src/utils/staking/index.ts
- src/utils/staking/param.ts
- src/utils/utxo/findInputUTXO.ts
- src/utils/utxo/getPsbtInputFields.ts
- src/utils/utxo/getScriptType.ts

Repository: babylonlabs-io/covenant-emulator

Audited Commit: c75e6de5039a661489a337a69580c0ae26823c35

Final Commit: 14a045a21798f07be15c2028ff49b974fe3388b5

Files:

- clientcontroller/babylon.go
- clientcontroller/interface.go
- clientcontroller/retry_utils.go
- cmd/covd/flags.go
- cmd/covd/init.go
- cmd/covd/key.go

- cmd/covd/main.go
- cmd/covd/start.go
- codec/codec.go
- config/babylon.go
- config/config.go
- config/metrics.go
- covenant/covenant.go
- covenant/expected_signer.go
- covenant/metrics.go
- covenant/service/prometheus.go
- covenant/service/server.go
- itest/babylon_node_handler.go
- itest/utils.go
- keyring/keyring.go
- keyring/keyringcontroller.go
- log/log.go
- testutil/datagen.go
- testutil/mocks/babylon.go
- testutil/utils.go
- tools/go.mod
- tools/go.sum
- tools/tools.go
- types/chainkey.go
- types/delegation.go
- types/params.go
- types/sigs.go
- types/txresponse.go
- util/path.go

Repository: babylonlabs-io/finality-provider

Audited Commit: 4857d562c6613dccbe56a55ca29c7626a61fb362

Final Commit: cb1c05ee7e349844916aeef521b3f61234b998b4

Files:

- clientcontroller/babylon.go
- clientcontroller/interface.go
- clientcontroller/retry_utils.go
- codec/codec.go
- eotsmanager/client/rpcclient.go
- eotsmanager/cmd/eotsd/main.go
- eotsmanager/config/config.go
- eotsmanager/config/db.go
- eotsmanager/eotsmanager.go
- eotsmanager/localmanager.go
- eotsmanager/proto/eotsmanager.proto
- eotsmanager/randgenerator/randgenerator.go
- eotsmanager/service/rpcserver.go
- eotsmanager/service/server.go
- eotsmanager/store/eotsstore.go
- eotsmanager/store/errors.go
- eotsmanager/types/errors.go
- eotsmanager/types/val_record.go
- finality-provider/cmd/cmd.go
- finality-provider/cmd/fpd/daemon/daemon_commands.go
- finality-provider/cmd/fpd/daemon/flags.go
- finality-provider/cmd/fpd/daemon/init.go
- finality-provider/cmd/fpd/daemon/keys.go
- finality-provider/cmd/fpd/daemon/start.go
- finality-provider/cmd/fpd/main.go
- finality-provider/config/babylon.go
- finality-provider/config/config.go
- finality-provider/config/db.go
- finality-provider/config/poller.go

- finality-provider/proto/finality_providers.go
- finality-provider/proto/finality_providers.proto
- finality-provider/proto/scripts/protocgen.sh
- finality-provider/service/app.go
- finality-provider/service/chain_poller.go
- finality-provider/service/client/rpcclient.go
- finality-provider/service/eots_manager_adapter.go
- finality-provider/service/errors.go
- finality-provider/service/fp_instance.go
- finality-provider/service/fp_store_adapter.go
- finality-provider/service/pub_rand_store_adapter.go
- finality-provider/service/rpcserver.go
- finality-provider/service/server.go
- finality-provider/store/errors.go
- finality-provider/store/fpstore.go
- finality-provider/store/pub_rand.go
- finality-provider/store/storedfp.go
- itest/container/config.go
- itest/container/container.go
- itest/eotsmanager_handler.go
- itest/utils.go
- keyring/keyring.go
- keyring/keyringcontroller.go
- log/log.go
- metrics/config.go
- metrics/eots_collectors.go
- metrics/fp_collectors.go
- metrics/server.go
- types/blockinfo.go
- types/chainkey.go
- types/pub_rand_commit.go

- types/stakingparams.go
- types/txresponse.go
- util/path.go
- version/version.go

Repository: babylonlabs-io/simple-staking

Audited Commit: 56eecdf1d5307020d5fcdd1b8a802761d64c82a3

Final Commit: 56eecdf1d5307020d5fcdd1b8a802761d64c82a3

Files:

- src/app/api/apiWrapper.ts
- src/app/api/error/index.ts
- src/app/api/getDelegations.ts
- src/app/api/getDelegationsV2.ts
- src/app/api/getFinalityProviders.ts
- src/app/api/getFinalityProvidersV2.ts
- src/app/api/getNetworkInfo.ts
- src/app/api/getStats.ts
- src/app/api/getSystemStats.ts
- src/app/api/getUnbondingEligibility.ts
- src/app/api/healthCheckClient.ts
- src/app/api/postFilterOrdinals.ts
- src/app/api/postUnbonding.ts
- src/app/components/Banner/Banner.tsx
- src/app/components/Container/Container.tsx
- src/app/components/Delegations/Activity.tsx
- src/app/components/Delegations/Delegation.tsx
- src/app/components/Delegations/DelegationActions.tsx
- src/app/components/Delegations/Delegations.tsx
- src/app/components/Delegations/components/DelegationCell.tsx
- src/app/components/Error/GenericError.tsx
- src/app/components/FAQ/FAQ.tsx

- src/app/components/FAQ/Section.tsx
- src/app/components/FAQ/data/questions.tsx
- src/app/components/Footer/Footer.tsx
- src/app/components/Hash/Hash.tsx
- src/app/components/Header/Header.tsx
- src/app/components/Header/SimplifiedHeader.tsx
- src/app/components/Loading/Loading.tsx
- src/app/components/Logo/Icon.tsx
- src/app/components/Logo/Logo.tsx
- src/app/components/Meta/MetaTags.tsx
- src/app/components/Modals/CancelFeedbackModal.tsx
- src/app/components/Modals/ClaimRewardModal.tsx
- src/app/components/Modals/ConfirmationModal.tsx
- src/app/components/Modals/ErrorModal.tsx
- src/app/components/Modals/InfoModal.tsx
- src/app/components/Modals/Phase2Here.tsx
- src/app/components/Modals/PreviewModal.tsx
- src/app/components/Modals/RegistrationModal/RegistrationEndModal.tsx
- src/app/components/Modals/RegistrationModal/RegistrationStartModal.tsx
- src/app/components/Modals/ResponsiveDialog.tsx
- src/app/components/Modals/SignModal/SignModal.tsx
- src/app/components/Modals/SignModal/Step.tsx
- src/app/components/Modals/SlashingModal.tsx
- src/app/components/Modals/StakeModal.tsx
- src/app/components/Modals/SubmitModal.tsx
- src/app/components/Modals/SuccessFeedbackModal.tsx
- src/app/components/Modals/UnbondModal.tsx
- src/app/components/Modals/VerificationModal.tsx
- src/app/components/Modals/WalletDisconnectModal.tsx
- src/app/components/Modals/WithdrawModal.tsx
- src/app/components/NetworkBadge/NetworkBadge.tsx

- src/app/components/NetworkBadge/testnet-icon.png
- src/app/components/Notification/DetailsButton.tsx
- src/app/components/Notification/FloatingTopBar.tsx
- src/app/components/Notification/IconWrapper.tsx
- src/app/components/Notification/Notification.tsx
- src/app/components/Notification/NotificationContainer.tsx
- src/app/components/Notification/NotificationText.tsx
- src/app/components/Notification/NotificationTitle.tsx
- src/app/components/PersonalBalance/PersonalBalance.tsx
- src/app/components/Section/Section.tsx
- src/app/components/Stakers/Staker.tsx
- src/app/components/Staking/FinalityProviders/FinalityProviderColumns.tsx
- src/app/components/Staking/FinalityProviders/FinalityProviderFilter.tsx
- src/app/components/Staking/FinalityProviders/FinalityProviderSearch.tsx
- src/app/components/Staking/FinalityProviders/FinalityProviderTable.tsx
- src/app/components/Staking/FinalityProviders/FinalityProviderTableStatusView.tsx
- src/app/components/Staking/FinalityProviders/FinalityProviders.tsx
- src/app/components/Staking/FinalityProviders/components/FPInfo.tsx
- src/app/components/Staking/Form/States/Message.tsx
- src/app/components/Staking/Form/States/StakingNotAvailable.tsx
- src/app/components/Staking/Form/States/WalletNotConnected.tsx
- src/app/components/Staking/Form/States/api-not-available.svg
- src/app/components/Staking/Form/States/connect-icon.svg
- src/app/components/Staking/Form/States/geo-restricted.svg
- src/app/components/Staking/Form/States/staking-cap-reached.svg
- src/app/components/Staking/Form/States/staking-not-started.svg
- src/app/components/Staking/Form/States/staking-upgrading.svg
- src/app/components/Staking/Form/States/wallet-icon.svg
- src/app/components/Staking/Form/validation/validation.ts
- src/app/components/Staking/StakingForm.tsx
- src/app/components/Stats/ActionComponent.tsx

- src/app/components/Stats/StatItem.tsx
- src/app/components/Stats/Stats.tsx
- src/app/components/Stats/icons/index.tsx
- src/app/components/ThemeToggle/ThemeToggle.tsx
- src/app/components/Wallet/Connect.tsx
- src/app/constants.ts
- src/app/constants/endpoints.ts
- src/app/constants/errorMessages.ts
- src/app/constants/index.ts
- src/app/context/Error/ErrorProvider.tsx
- src/app/context/Error/errors/clientError.ts
- src/app/context/Error/errors/index.ts
- src/app/context/Error/errors/serverError.ts
- src/app/context/api/StakingStatsProvider.tsx
- src/app/context/rpc/BbnRpcProvider.tsx
- src/app/context/tomo/BBNConnector.tsx
- src/app/context/tomo/BTCConnector.tsx
- src/app/context/tomo/TomoProvider.tsx
- src/app/context/tomo/TomoWidget.tsx
- src/app/context/tomo/tomo.png
- src/app/context/wallet/BTCWalletProvider.tsx
- src/app/context/wallet/CosmosWalletProvider.tsx
- src/app/context/wallet/WalletConnectionProvider.tsx
- src/app/global-error.tsx
- src/app/hooks/client/api/useBTCBalance.ts
- src/app/hooks/client/api/useDelegations.ts
- src/app/hooks/client/api/useDelegationsV2.ts
- src/app/hooks/client/api/useFinalityProviders.ts
- src/app/hooks/client/api/useFinalityProvidersV2.ts
- src/app/hooks/client/api/useNetworkFees.ts
- src/app/hooks/client/api/useNetworkInfo.ts

- src/app/hooks/client/api/useOrdinals.ts
- src/app/hooks/client/api/useSystemStats.ts
- src/app/hooks/client/api/useUTXOs.ts
- src/app/hooks/client/rpc/mutation/useBbnTransaction.ts
- src/app/hooks/client/rpc/mutation/useSigningStargateClient.ts
- src/app/hooks/client/rpc/queries/useBbnQuery.ts
- src/app/hooks/client/useClient.ts
- src/app/hooks/services/useDelegationService.ts
- src/app/hooks/services/useRegistrationService.ts
- src/app/hooks/services/useRewardsService.ts
- src/app/hooks/services/useStakingManagerService.ts
- src/app/hooks/services/useStakingService.ts
- src/app/hooks/services/useTransactionService.ts
- src/app/hooks/services/useV1TransactionService.ts
- src/app/hooks/storage/useDelegationStorage.ts
- src/app/hooks/useBreakpoint.ts
- src/app/hooks/useHealthCheck.ts
- src/app/hooks/useNotification.tsx
- src/app/layout.tsx
- src/app/not-found.tsx
- src/app/page.tsx
- src/app/providers.tsx
- src/app/services/healthCheckService.ts
- src/app/state/BalanceState.tsx
- src/app/state/DelegationState.tsx
- src/app/state/DelegationV2State.tsx
- src/app/state/FinalityProviderState.tsx
- src/app/state/RewardState.tsx
- src/app/state/StakingState.tsx
- src/app/state/index.tsx
- src/app/types/api.ts

- src/app/types/api/healthCheck.ts
- src/app/types/delegations.ts
- src/app/types/delegationsV2.ts
- src/app/types/errors.ts
- src/app/types/fee.ts
- src/app/types/finalityProviders.ts
- src/app/types/network.ts
- src/app/types/networkInfo.ts
- src/app/types/services/healthCheck.ts
- src/app/types/stakingParams.ts
- src/app/types/stakingStats.ts
- src/components/common/AuthGuard/index.tsx
- src/components/common/Body/index.tsx
- src/components/common/GridTable/components/TCell.tsx
- src/components/common/GridTable/components/THead.tsx
- src/components/common/GridTable/index.tsx
- src/components/common/GridTable/types.ts
- src/components/common/GridTable/utils.ts
- src/components/common/Hint/index.tsx
- src/components/delegations/DelegationList/components/ActionButton.tsx
- src/components/delegations/DelegationList/components/Amount.tsx
- src/components/delegations/DelegationList/components/DelegationModal.tsx
- src/components/delegations/DelegationList/components/FinalityProviderMoniker.tsx
- src/components/delegations/DelegationList/components/Inception.tsx
- src/components/delegations/DelegationList/components/Overflow.tsx
- src/components/delegations/DelegationList/components/SlashingContent.tsx
- src/components/delegations/DelegationList/components/Status.tsx
- src/components/delegations/DelegationList/components/TxHash.tsx
- src/components/delegations/DelegationList/index.tsx
- src/components/delegations/DelegationList/utils.ts
- src/components/staking/StakingForm/components/AmountField.tsx

- src/components/staking/StakingForm/components/BBNFeeAmount.tsx
- src/components/staking/StakingForm/components/BTCFeeAmount.tsx
- src/components/staking/StakingForm/components/BTCFeeRate.tsx
- src/components/staking/StakingForm/components/FeeItem.tsx
- src/components/staking/StakingForm/components/FeeSection.tsx
- src/components/staking/StakingForm/components/InfoAlert.tsx
- src/components/staking/StakingForm/components/Overlay.tsx
- src/components/staking/StakingForm/components/SubmitButton.tsx
- src/components/staking/StakingForm/components/TermField.tsx
- src/components/staking/StakingForm/components/Total.tsx
- src/components/staking/StakingForm/index.tsx
- src/components/staking/StakingModal/index.tsx
- src/config/index.ts
- src/config/network/bbn.ts
- src/config/network/bbn/canary.ts
- src/config/network/bbn/devnet.ts
- src/config/network/btc.ts
- src/config/network/index.ts
- src/config/screen-breakpoints.ts
- src/hooks/useCurrentTime.tsx
- src/utils/bbn.ts
- src/utils/btc.ts
- src/utils/buffer.ts
- src/utils/chunkArray.ts
- src/utils/createStateUtils.ts
- src/utils/delegations/fee.ts
- src/utils/delegations/index.ts
- src/utils/delegations/slashing.ts
- src/utils/fetch.ts
- src/utils/getFeeRateFromMempool.ts
- src/utils/getStakingTerm.ts

- src/utils/getState.ts
- src/utils/index.ts
- src/utils/isStakingSignReady.ts
- src/utils/local_storage/calculateDelegationsDiff.ts
- src/utils/local_storage/filterDelegationsLocalStorage.ts
- src/utils/local_storage/getDelegationsLocalStorageKey.ts
- src/utils/local_storage/getIntermediateDelegationsLocalStorageKey.ts
- src/utils/local_storage/toLocalStorageIntermediateDelegation.ts
- src/utils/maxDecimals.ts
- src/utils/mempool_api.ts
- src/utils/nextPowerOfTwo.ts
- src/utils/params/index.ts
- src/utils/time.ts
- src/utils/trim.ts
- src/utils/types.ts
- src/utils/url.ts
- src/utils/version.ts
- src/utils/wallet/bbnRegistry.ts
- src/utils/wallet/errors.ts
- src/utils/wallet/icons/bitget.svg
- src/utils/wallet/icons/cactuslink.svg
- src/utils/wallet/icons/keystone.svg
- src/utils/wallet/icons/okx.svg
- src/utils/wallet/icons/onekey.svg
- src/utils/wallet/icons/tomo.svg
- src/utils/wallet/index.ts

Repository: babylonlabs-io/staking-api-service

Audited Commit: 3f4cff4c91398d2737ab66a38dad40303aca1443

Final Commit: 1956e122cf31fb5605dffefa42b7b3a0ea9bf404

Files:

- cmd/staking-api-service/cli/root.go
- cmd/staking-api-service/main.go
- cmd/staking-api-service/scripts/pubkey_address_backfill.go
- cmd/staking-api-service/scripts/replay_unprocessed_messages.go
- internal/indexer/db/client/db_client.go
- internal/indexer/db/client/delegation.go
- internal/indexer/db/client/finality_provider.go
- internal/indexer/db/client/interface.go
- internal/indexer/db/client/last_processed_height.go
- internal/indexer/db/client/params.go
- internal/indexer/db/model/delegation.go
- internal/indexer/db/model/finality_providers.go
- internal/indexer/db/model/last_processed_height.go
- internal/indexer/db/model/params.go
- internal/indexer/db/model/setup.go
- internal/indexer/db/model/timelock.go
- internal/indexer/types/bbn_params.go
- internal/indexer/types/delegation_state.go
- internal/shared/api/handlers/handler/handler.go
- internal/shared/api/handlers/handler/health.go
- internal/shared/api/handlers/handler/ordinals.go
- internal/shared/api/handlers/handlers.go
- internal/shared/api/http_response.go
- internal/shared/api/middlewares/content_length.go
- internal/shared/api/middlewares/cors.go
- internal/shared/api/middlewares/logging.go
- internal/shared/api/middlewares/security.go
- internal/shared/api/middlewares/tracing.go
- internal/shared/api/routes.go
- internal/shared/api/server.go
- internal/shared/config/assets.go

- internal/shared/config/config.go
- internal/shared/config/db.go
- internal/shared/config/delegation_transition.go
- internal/shared/config/metrics.go
- internal/shared/config/ordinals.go
- internal/shared/config/server.go
- internal/shared/db/client/db_client.go
- internal/shared/db/client/interface.go
- internal/shared/db/client/pk_address_mapping.go
- internal/shared/db/client/unprocessable_message.go
- internal/shared/db/clients/db_clients.go
- internal/shared/db/error.go
- internal/shared/db/model/pagination.go
- internal/shared/db/model/pk_address_mapping.go
- internal/shared/db/model/setup.go
- internal/shared/db/model/unprocessable_message.go
- internal/shared/db/pagination.go
- internal/shared/http/client/http_client.go
- internal/shared/http/client/interface.go
- internal/shared/http/clients/http_clients.go
- internal/shared/http/clients/ordinals/interface.go
- internal/shared/http/clients/ordinals/ordinals.go
- internal/shared/observability/healthcheck/healthcheck.go
- internal/shared/observability/metrics/metrics.go
- internal/shared/observability/tracing/tracing.go
- internal/shared/services/service/interface.go
- internal/shared/services/service/ordinals.go
- internal/shared/services/service/service.go
- internal/shared/services/services.go
- internal/shared/types/delegation.go
- internal/shared/types/error.go

- internal/shared/types/finality_providers.go
- internal/shared/types/global_params.go
- internal/shared/types/staker.go
- internal/shared/types/transaction_type.go
- internal/shared/types/utxo.go
- internal/shared/utils/btc.go
- internal/shared/utils/datagen/datagen.go
- internal/shared/utils/state_transition.go
- internal/shared/utils/timestamp.go
- internal/shared/utils/utils.go
- internal/shared/utils/validation.go
- internal/v1/api/handlers/delegation.go
- internal/v1/api/handlers/finality_provider.go
- internal/v1/api/handlers/handler.go
- internal/v1/api/handlers/params.go
- internal/v1/api/handlers/pubkey.go
- internal/v1/api/handlers/staker.go
- internal/v1/api/handlers/stats.go
- internal/v1/api/handlers/unbonding.go
- internal/v1/db/client/btc_info.go
- internal/v1/db/client/db_client.go
- internal/v1/db/client/delegation.go
- internal/v1/db/client/interface.go
- internal/v1/db/client/stats.go
- internal/v1/db/client/timelock.go
- internal/v1/db/client/unbonding.go
- internal/v1/db/client/withdraw.go
- internal/v1/db/model/btc_info.go
- internal/v1/db/model/delegation.go
- internal/v1/db/model/stats.go
- internal/v1/db/model/timelock.go

- internal/v1/db/model/unbonding.go
- internal/v1/service/delegation.go
- internal/v1/service/finality_provider.go
- internal/v1/service/interface.go
- internal/v1/service/params.go
- internal/v1/service/service.go
- internal/v1/service/staker.go
- internal/v1/service/stats.go
- internal/v1/service/timelock.go
- internal/v1/service/unbonding.go
- internal/v1/service/withdrawn.go
- internal/v2/api/handlers/delegation.go
- internal/v2/api/handlers/finality_provider.go
- internal/v2/api/handlers/handler.go
- internal/v2/api/handlers/network.go
- internal/v2/api/handlers/stats.go
- internal/v2/db/client/db_client.go
- internal/v2/db/client/interface.go
- internal/v2/db/client/stats.go
- internal/v2/db/model/stats.go
- internal/v2/queue/handler/handler.go
- internal/v2/queue/handler/stats.go
- internal/v2/queue/queue.go
- internal/v2/service/delegation.go
- internal/v2/service/finality_provider.go
- internal/v2/service/interface.go
- internal/v2/service/network.go
- internal/v2/service/network_test.go
- internal/v2/service/params.go
- internal/v2/service/service.go
- internal/v2/service/staker.go

- internal/v2/service/stats.go
- internal/v2/types/delegation_states.go

Repository: babylonlabs-io/staking-expiry-checker

Audited Commit: c46282e02cf857b2f3ce69abf9132b3415097d39

Final Commit: bae0105076808a5721ad48ca0446416eaa6a8749

Files:

- cmd/staking-expiry-checker/cli/root.go
- cmd/staking-expiry-checker/main.go
- internal/btcclient/btcclient.go
- internal/btcclient/interface.go
- internal/btcclient/notifier.go
- internal/config/btc.go
- internal/config/db.go
- internal/config/metrics.go
- internal/config/poller.go
- internal/db/dbclient.go
- internal/db/delegation.go
- internal/db/error.go
- internal/db/expiry.go
- internal/db/interface.go
- internal/db/model/delegation.go
- internal/db/model/pagination.go
- internal/db/model/setup.go
- internal/db/model/timelock.go
- internal/observability/metrics/metrics.go
- internal/observability/tracing/tracing.go
- internal/services/delegation_handlers.go
- internal/services/pollers.go
- internal/services/service.go
- internal/services/tracked_subscriptions.go

- internal/services/watch_btc_events.go
- internal/types/delegation.go
- internal/types/delegation_events.go
- internal/types/error.go
- internal/types/global_params.go
- internal/types/transaction.go
- internal/utils/state_transition.go
- internal/utils/utils.go
- params/params.go

Repository: babylonlabs-io/staking-queue-client

Audited Commit: c4b08ada1f40852bdeecee550866eae308b52616

Final Commit: 6b9bb1d59a7d6c5c19ab534f705cd7a5d61ebf91

Files:

- client/client.go
- client/rabbitmq_client.go
- client/schema.go
- config/queue.go
- queuemngr/queue_manager.go

Repository: babylonlabs-io/vigilante

Audited Commit: 33325f3eb8c897508b47ab6e729b03877065ec2f

Final Commit: 7cf12f431d178020fef840e3a79522bd86134c8d

Files:

- btcclient/client.go
- btcclient/client_wallet.go
- btcclient/interface.go
- btcclient/notifier.go
- btcclient/query.go
- btcstaking-tracker/atomicslasher/atomic_slasher.go
- btcstaking-tracker/atomicslasher/babylon_adapter.go

- btcstaking-tracker/atomicslasher/expected_babylon_client.go
- btcstaking-tracker/atomicslasher/routines.go
- btcstaking-tracker/atomicslasher/types.go
- btcstaking-tracker/btcslasher/bootstrapping.go
- btcstaking-tracker/btcslasher/btc_utils.go
- btcstaking-tracker/btcslasher/expected_babylon_client.go
- btcstaking-tracker/btcslasher/slasher.go
- btcstaking-tracker/btcslasher/slasher_utils.go
- btcstaking-tracker/expected_routines.go
- btcstaking-tracker/stakingeventwatcher/expected_babylon_client.go
- btcstaking-tracker/stakingeventwatcher/stakingeventwatcher.go
- btcstaking-tracker/stakingeventwatcher/tracked_delegations.go
- btcstaking-tracker/tracker.go
- cmd/vigilante/cmd/btcstaking_tracker.go
- cmd/vigilante/cmd/monitor.go
- cmd/vigilante/cmd/reporter.go
- cmd/vigilante/cmd/root.go
- cmd/vigilante/cmd/submitter.go
- cmd/vigilante/cmd/utils.go
- cmd/vigilante/main.go
- config/bitcoin.go
- config/btcstaking_tracker.go
- config/common.go
- config/config.go
- config/dbconfig.go
- config/grpc.go
- config/grpcweb.go
- config/log.go
- config/metrics.go
- config/monitor.go
- config/reporter.go

- config/submitter.go
- metrics/btcstaking_tracker.go
- metrics/monitor.go
- metrics/prometheus.go
- metrics/reporter.go
- metrics/submitter.go
- monitor/btcscanner/block_handler.go
- monitor/btcscanner/btc_scanner.go
- monitor/expected_babylon_client.go
- monitor/liveness_checker.go
- monitor/monitor.go
- monitor/query.go
- monitor/store/store.go
- monitor/utils.go
- netparams/bitcoin.go
- reporter/block_handler.go
- reporter/bootstrapping.go
- reporter/expected_babylon_client.go
- reporter/reporter.go
- reporter/utils.go
- retrywrap/retry.go
- rpcserver/server.go
- rpcserver/service.go
- rpcserver/tls.go
- submitter/expected_babylon_client.go
- submitter/poller/expected_babylon_client.go
- submitter/poller/poller.go
- submitter/relayer/change_address.go
- submitter/relayer/estimator.go
- submitter/relayer/relayer.go
- submitter/relayer/utils.go

- submitter/store/store.go
- submitter/submitter.go
- testutil/datagen/datagen.go
- testutil/datagen/reporter.go
- testutil/mocks/btcclient.go
- testutil/port.go
- testutil/store.go
- testutil/version.go
- types/btccache.go
- types/btccheckpoint.go
- types/btclightclient.go
- types/ckpt.go
- types/ckpt_bookkeeper.go
- types/ckpt_cache.go
- types/ckpt_info.go
- types/ckpt_record.go
- types/ckpt_segment.go
- types/epoch_info.go
- types/errors.go
- types/genesis_info.go
- types/indexed_block.go
- types/safeprivatekey.go
- types/utils.go
- types/utxo.go
- utils/serialize.go

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.