5.0 Technical Analysis

Medium

Issue#1 [Resolved]

[M-1] Freezing of Validator Bond Delegations

Severity

Medium

Issue Description

A flaw in the interaction between **tokenized shares, validator bond shares, and delegation mechanisms** within the staking module allows an attacker to **manipulate ValidatorBondShares**, potentially disrupting other delegators.

The issue arises because:

1. **Tokenized shares cannot be marked as validator bond**, but it is still possible to **transfer tokens to a delegator with an existing validator bond delegation**.

2. The **RedeemTokensForShares function** currently permits validator bond delegations but **fails to update ValidatorBondShares** accordingly.

Exploitation Scenario

- A malicious delegator can delegate a small amount and mark it as ValidatorBond.
- They then convert this into a large number of LSM tokens, redeem them, and undelegate the funds.
- This artificially decreases the validator's ValidatorBondShares.
- As a result, other delegators with ValidatorBond delegations to the same validator may become unable to undelegate their funds, since SafelyDecreaseValidatorBond would fail due to insufficient ValidatorBondShares.
- This attack requires minimal capital but can cause significant disruptions in the staking process.

Issue Impact

Freezing of Validator Bond Delegations:

- Delegators with validator bond delegations **may be unable to undelegate their funds** due to insufficient ValidatorBondShares.
- This disrupts the normal **staking and unstaking process**, affecting liquidity for affected delegators.

Manipulation of ValidatorBondShares:

- A malicious actor can **artificially reduce** a validator's ValidatorBondShares with minimal capital.
- This impacts other delegators who rely on validator bond delegations, creating an unfair staking environment.

Potential Staking Disruptions:

- The exploit could **destabilize validator operations**, leading to staking inefficiencies or unintended penalties.
- Affected delegators may **lose confidence in the staking mechanism**, reducing overall participation.

Economic and Security Risks:

- Attackers can **intentionally disrupt validator bonding**, causing potential harm to the network's integrity.
- If widely exploited, this could **discourage users from participating in validator bonding**, weakening the staking security model.

Recommended Mitigation Steps

Enhance Redemption Validation in RedeemTokensForShares:

- Implement a **check to determine if the destination delegation is a validator bond** before allowing redemption.
- Either **block the redemption process entirely** or **ensure that ValidatorBondShares are correctly updated** to reflect the changes.

.

Issue#2 [Acknowledged]

[M-2] Lack of Liquid Staking Accounting in CreateValidator

Severity

Medium

Issue Description

The **CreateValidator function** does not incorporate the **liquid staking accounting mechanisms** present in the **Delegate function**.

In the **Delegate function**, specific checks and updates are performed when a **liquid staking provider initiates a delegation**:

- Staked tokens are converted into an **equivalent number of shares** in the validator.

- The **global liquid stake** and **validator liquid shares** are **updated accordingly** to maintain correct accounting.

However, during **validator creation**, when the initial stake is self-delegated:

- **No such checks or updates occur** for liquid staking.

- This results in **inconsistent tracking**, as the **global liquid staking cap and per-validator cap remain unchanged**.

- This creates a potential discrepancy between the actual **liquid stake in the system** and what is recorded, leading to potential **accounting and governance issues**.

Issue Impact

Inconsistent Liquid Staking Limits:

- The global and per-validator liquid staking caps do not reflect the actual liquid stake amounts.
- This could lead to violations of staking constraints, potentially affecting network security.

Potential Exploits in Liquid Staking Systems:

- Validators could circumvent staking caps, enabling over-exposure of liquid stake beyond intended limits.
- This could lead to imbalanced validator participation and unintended dominance by certain validators.

Discrepancies in Staking Accounting:

- The network may fail to enforce liquid staking limits properly, allowing

incorrect stake distributions.
- This could lead to unexpected behavior in governance, slashing mechanisms, or stake-weighted decisions.

Recommended Mitigation Steps

Integrate Liquid Staking Bookkeeping into CreateValidator:

- Ensure that **the same accounting checks and updates from Delegate** are applied when creating a validator.
- Convert the **initial self-stake into shares properly** and **update the global and per-validator liquid staking caps accordingly**.

Issue#3 [Resolved]

[M-3] Inconsistencies in Slash Redelegation

Severity

Medium

Issue Description

A flaw in the **SlashRedelegation** function leads to inconsistencies when handling **redelegated stakes involving a validator bond**.

In the **Cosmos SDK**, when a validator is slashed for an infraction:

- **Penalties apply to both direct delegations and redelegations**.
- **Redelegated tokens are penalized** by **unbonding an equivalent slash amount**.

Issues Identified:

1. **Validator Bond Shares Not Adjusted:**
    - If the redelegated stake involves a **validator bond**, the **validatorBondShare** should also be reduced.
    - Since the validatorBond represents the **validator's self-staked commitment**, penalties must also **impact this bond** when slashing occurs.
    - **Failing to adjust validatorBondShare** leaves the **validator's total staked shares artificially inflated**.
2. **LiquidShares Not Updated After Slashing:**
    - If the **delegatorAddress** is a **liquid staker**, the **Unbond function** is triggered with `sharesToUnbond`.
    - This **reduces the redelegated balance** but **does not update LiquidShares** in the validator's record.
    - As a result, the **validator's liquid stake remains artificially high**, leading to **incorrect staking calculations**.

```
func (k Keeper) SlashRedelegation(ctx context.Context, srcValidator types.Validator,
    |→ redelegation types.Redelegation,
    infractionHeight int64, slashFactor math.LegacyDec,
) (totalSlashAmount math.Int, err error) {
    [...]
    tokensToBurn, err := k.Unbond(ctx, delegatorAddress, valDstAddr, sharesToUnbond)
    if err != nil {
             return math.ZeroInt(), err
    }
     [...]
}
```

Issue Impact

Validator Bond Inflation:

- Validators may retain more self-staked shares than they should, leading to an overstated commitment.
- This weakens the security model, as validators appear to be staking more than they actually are.

Liquid Staking Inconsistencies:

- Slashed liquid stake is not properly reflected, making staking data unreliable.
- This could allow incorrect governance weight calculations and distort stake-based rewards.

Potential for Exploitation:

- A validator could retain an artificially high stake post-slashing, potentially avoiding penalties they should incur.
- Liquid stakers may not properly account for slashed amounts, leading to incorrect economic balances.

Recommended Mitigation Steps

Reduce validatorBondShare When Slashing Occurs

- Ensure that **when a validator bond is involved in a redelegation**, slashing appropriately reduces **validatorBondShare**.

Update LiquidShares When Unbonding Liquid Stake

- Modify `SlashRedelegation` to correctly **adjust the LiquidShares value** in the validator’s records when slashed.

Low

Issue#4 [Resolved]
[L-1] Improper Storage Utilization

Severity
Low

Issue Description
msg_server::TokenizeShares does not ensure that the share is not zero. It is possible for the amount to be positive while the share is zero, resulting in the creation and storage of records without any actual shares.

```
func (k msgServer) TokenizeShares(goCtx context.Context, msg
*types.MsgTokenizeShares)
    |→ (*types.MsgTokenizeSharesResponse, error) {
    [...]
    shares, err := k.ValidateUnbondAmount(
                ctx, delegatorAddress, valAddr, msg.Amount.Amount,
        )
        if err != nil {
            return nil, err
        }
   [...]
}
```

Recommended Mitigation Steps
Add a check after the call to **ValidateUnbondAmount** to ensure that **shares** is greater than zero.

Issue#5 [Resolved]

[L-2] Insufficient Error Handling

Severity

Low

Issue Description

ValidatorBond , **RedeemTokensForShares** , and **UnbondValidator** in **msg_server** lack sufficient error handling for certain function calls (**SetDelegation** , **bondedTokensToNotBonded** , and **jailValidator** , respectively). Missing error checks may affect delegation states in the staking module, leading to inconsistencies in the bonded and not-bonded pools and impacting validator management.

Recommended Mitigation Steps

Utilize a linter, such as **errcheck** , which checks for unchecked errors in Go code. **errcheck** helps detect cases where error handling is missing.