// Security Assessment

10.02.2025 - 10.30.2025

Babylon Genesis

Babylon Labs

HALBORN

Babylon Genesis - Babylon Labs

Prepared by: HALBORN

Last Updated 11/28/2025

Date of Engagement: October 2nd, 2025 - October 30th, 2025

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
9	0	0	0	2	7

TABLE OF CONTENTS

1. Introduction
2. Assessment summary
3. Test approach and methodology
4. Risk methodology
5. Scope
6. Assessment summary & findings overview
7. Findings & Tech Details
   - 7.1 Missing unbonding/redelegation entry limit checks in queued epoching messages
   - 7.2 Msgwrappedcreatevalidator creates untrackable queued messages due to missing txid and msgid fields
   - 7.3 Unbounded costaker iteration in msgupdateparams enables dos
   - 7.4 Proof of possession does not use domain separator for the signature
   - 7.5 Jailed finality providers can receive delegations
   - 7.6 Deleted finality providers can receive delegations
   - 7.7 Misleading internal key handling in derivetaprootpkscript
   - 7.8 Duplicate staking outputs slip past validation
   - 7.9 Delegation transaction op_return is not verified

1. Introduction

`Babylon` engaged `Halborn` to perform a security assessment of their core chain and peripheral services from October 2nd, 2025 to October 30th, 2025. The assessment scope was limited to the repositories provided to Halborn. Commit hashes and additional details are available in the Scope section of this report.

2. Assessment Summary

`Halborn` was allocated 4 weeks for this engagement and assigned 1 full-time security engineer to conduct a comprehensive review of the assets within scope. The engineer is an expert in blockchain and smart contract security, with advanced skills in penetration testing and smart contract exploitation, as well as extensive knowledge of multiple blockchain protocols.

The objectives of this assessment are to:

- Identify potential security vulnerabilities within the core chain.
- Verify that the functionalities operates as intended.

In summary, Halborn identified several areas for improvement to reduce the likelihood and impact of security risks, which were addressed by the `Babylon team`. All issues were resolved by the Babylon team, except for informational severity findings that were acknowledged as intended design choice. The primary recommendations were:

- `Process costaker score recalculations in fixed size batches, rather than iterating over the entire set in a single governance-triggered transaction.`
- `Utilize the HasMaxUnbondingDelegationEntries and HasMaxRedelegationEntries helpers before enqueuing the message.`
- `Construct the QueuedMessage using the standard helper.`

3. Test Approach And Methodology

`Halborn` conducted a combination of manual code review and automated security testing to balance efficiency, timeliness, practicality, and accuracy within the scope of this assessment. While manual testing is crucial for identifying flaws in logic, processes, and implementation, automated testing enhances coverage of the codebase and quickly detects deviations from established security best practices.

The following phases and associated tools were employed throughout the term of the assessment:

- Research into the platform's architecture, purpose and use.
- Manual code review and walkthrough of the codebase to identify any logical issues.
- Comprehensive assessment of the safety and usage of critical Golang variables and functions within scope that could lead to arithmetic-related vulnerabilities.
- Local testing using custom scripts and unit tests.
- Static security analysis.

4. RISK METHODOLOGY

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

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

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

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

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

4.1 EXPLOITABILITY

ATTACK ORIGIN (AO):

Captures whether the attack requires compromising a specific account.

ATTACK COST (AC):

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

ATTACK COMPLEXITY (AX):

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

METRICS:

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 IMPACT

CONFIDENTIALITY (C):

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

INTEGRITY (I):

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

AVAILABILITY (A):

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

DEPOSIT (D):

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

YIELD (Y):

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

METRICS:

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
Confidentiality (C)	None (C:N) Low (C:L) Medium (C:M) High (C:H) Critical (C:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

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

4.3 SEVERITY COEFFICIENT

REVERSIBILITY (R):

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

SCOPE (S):

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

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

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

The score is rounded up to 1 decimal places.

SEVERITY	SCORE VALUE RANGE
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4

SEVERITY	SCORE VALUE RANGE
Informational	0 - 1.9

5. SCOPE

REPOSITORY

(a) Repository: babylon-sdk

(b) Assessed Commit ID: 7e6c757

(c) Items in scope:

- x/btccheckpoint/README.md
- 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/genesis_test.go
- x/btccheckpoint/keeper/genesis.go
- x/btccheckpoint/keeper/genesis_test.go
- x/btccheckpoint/keeper/grpc_query.go
- x/btccheckpoint/keeper/grpc_query_params.go
- x/btccheckpoint/keeper/grpc_query_params_test.go
- x/btccheckpoint/keeper/grpc_query_test.go
- x/btccheckpoint/keeper/hooks.go
- x/btccheckpoint/keeper/keeper.go
- x/btccheckpoint/keeper/keeper_test.go
- x/btccheckpoint/keeper/msg_server.go
- x/btccheckpoint/keeper/msg_server_test.go
- x/btccheckpoint/keeper/params.go
- x/btccheckpoint/keeper/params_test.go
- x/btccheckpoint/keeper/submissions.go
- x/btccheckpoint/module.go
- x/btccheckpoint/types/btccheckpoint.pb.go
- x/btccheckpoint/types/btcutils.go
- x/btccheckpoint/types/btcutils_test.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/genesis.pb.go
- x/btccheckpoint/types/genesis_test.go
- x/btccheckpoint/types/incentive.go
- x/btccheckpoint/types/keys.go
- x/btccheckpoint/types/mock_keepers.go

- x/btccheckpoint/types/msgs.go
- x/btccheckpoint/types/params.go
- x/btccheckpoint/types/params.pb.go
- x/btccheckpoint/types/params_test.go
- x/btccheckpoint/types/query.go
- x/btccheckpoint/types/query.pb.go
- x/btccheckpoint/types/query.pb.gw.go
- x/btccheckpoint/types/tx.pb.go
- x/btccheckpoint/types/types.go
- x/btccheckpoint/types/types_test.go
- x/btclightclient/README.md
- x/btclightclient/client/cli/query.go
- x/btclightclient/client/cli/tx.go
- x/btclightclient/genesis.go
- x/btclightclient/genesis_test.go
- x/btclightclient/keeper/base_btc_header.go
- x/btclightclient/keeper/base_btc_header_test.go
- x/btclightclient/keeper/export_headers_state_test.go
- x/btclightclient/keeper/grpc_query.go
- x/btclightclient/keeper/grpc_query_test.go
- x/btclightclient/keeper/hooks.go
- x/btclightclient/keeper/keeper.go
- x/btclightclient/keeper/keeper_test.go
- x/btclightclient/keeper/msg_server.go
- x/btclightclient/keeper/msg_server_test.go
- x/btclightclient/keeper/params.go
- x/btclightclient/keeper/params_test.go
- x/btclightclient/keeper/state.go
- x/btclightclient/keeper/state_test.go
- x/btclightclient/keeper/triggers.go
- x/btclightclient/keeper/triggers_test.go
- x/btclightclient/keeper/utils.go
- x/btclightclient/keeper/utils_test.go
- x/btclightclient/module.go
- x/btclightclient/types/btc_header_info.go
- x/btclightclient/types/btc_header_info_test.go
- x/btclightclient/types/btc_light_client.go
- x/btclightclient/types/btclightclient.pb.go
- x/btclightclient/types/codec.go
- x/btclightclient/types/errors.go
- x/btclightclient/types/event.pb.go
- x/btclightclient/types/expected_keepers.go
- x/btclightclient/types/genesis.go
- x/btclightclient/types/genesis.pb.go

- x/btclightclient/types/genesis_test.go
- x/btclightclient/types/hooks.go
- x/btclightclient/types/keys.go
- x/btclightclient/types/keys_test.go
- x/btclightclient/types/msgs.go
- x/btclightclient/types/msgs_test.go
- x/btclightclient/types/params.go
- x/btclightclient/types/params.pb.go
- x/btclightclient/types/querier.go
- x/btclightclient/types/querier_test.go
- x/btclightclient/types/query.go
- x/btclightclient/types/query.pb.go
- x/btclightclient/types/query.pb.gw.go
- x/btclightclient/types/tx.pb.go
- x/btclightclient/types/types.go
- x/btclightclient/types/utils.go
- x/btclightclient/types/work.go
- x/btclightclient/types/work_test.go
- x/btcstaking/README.md
- 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/docs/btc-reorg.md
- x/btcstaking/docs/registration-eligibility.md
- x/btcstaking/docs/static/stakingtimeline.png
- x/btcstaking/genesis.go
- x/btcstaking/genesis_test.go
- x/btcstaking/keeper/allowed_transaction_index.go
- x/btcstaking/keeper/bench_test.go
- x/btcstaking/keeper/btc_delegations.go
- x/btcstaking/keeper/btc_delegators.go
- x/btcstaking/keeper/btc_height_index.go
- x/btcstaking/keeper/btc_height_index_test.go
- x/btcstaking/keeper/btc_reorg.go
- x/btcstaking/keeper/btc_reorg_test.go
- x/btcstaking/keeper/finality_providers.go
- x/btcstaking/keeper/finality_providers_test.go
- x/btcstaking/keeper/genesis.go
- x/btcstaking/keeper/genesis_test.go
- x/btcstaking/keeper/grpc_query.go
- x/btcstaking/keeper/grpc_query_test.go
- x/btcstaking/keeper/hooks.go

- x/btcstaking/keeper/hooks_test.go
- x/btcstaking/keeper/inclusion_proof.go
- x/btcstaking/keeper/inclusion_proof_test.go
- x/btcstaking/keeper/keeper.go
- x/btcstaking/keeper/msg_server.go
- x/btcstaking/keeper/msg_server_test.go
- x/btcstaking/keeper/params.go
- x/btcstaking/keeper/params_test.go
- x/btcstaking/keeper/power_dist_change.go
- x/btcstaking/keeper/query.go
- x/btcstaking/keeper/query_params.go
- x/btcstaking/keeper/query_params_test.go
- x/btcstaking/keeper/unbonding_transaction_sig_validation.go
- x/btcstaking/keeper/unbonding_transaction_sig_validation_test.go
- x/btcstaking/module.go
- x/btcstaking/types/btc_delegation.go
- x/btcstaking/types/btc_delegation_test.go
- x/btcstaking/types/btc_slashing_tx.go
- x/btcstaking/types/btc_slashing_tx_test.go
- x/btcstaking/types/btc_undelegation.go
- x/btcstaking/types/btc_undelegation_test.go
- x/btcstaking/types/btcstaking.go
- x/btcstaking/types/btcstaking.pb.go
- x/btcstaking/types/btcstaking_test.go
- x/btcstaking/types/codec.go
- x/btcstaking/types/commission.go
- x/btcstaking/types/create_delegation_parser.go
- x/btcstaking/types/errors.go
- x/btcstaking/types/events.go
- x/btcstaking/types/events.pb.go
- x/btcstaking/types/expected_keepers.go
- x/btcstaking/types/fuzz_create_finality_provider_signature_test.go
- x/btcstaking/types/genesis.go
- x/btcstaking/types/genesis.pb.go
- x/btcstaking/types/genesis_test.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/msg_test.go
- x/btcstaking/types/params.go
- x/btcstaking/types/params.pb.go
- x/btcstaking/types/params_test.go

- x/btcstaking/types/pop.go
- x/btcstaking/types/pop.pb.go
- x/btcstaking/types/pop_parity_poc_test.go
- x/btcstaking/types/pop_test.go
- x/btcstaking/types/query.go
- x/btcstaking/types/query.pb.go
- x/btcstaking/types/query.pb.gw.go
- x/btcstaking/types/tx.pb.go
- x/btcstaking/types/validate_parsed_message.go
- x/btcstaking/types/validate_parsed_message_test.go
- x/checkpointing/README.md
- x/checkpointing/abci.go
- x/checkpointing/client/cli/query.go
- x/checkpointing/client/cli/tx.go
- x/checkpointing/client/cli/tx_test.go
- x/checkpointing/client/cli/utils.go
- x/checkpointing/genesis.go
- x/checkpointing/genesis_test.go
- x/checkpointing/keeper/bls_signer.go
- x/checkpointing/keeper/bls_signer_test.go
- x/checkpointing/keeper/ckpt_state.go
- x/checkpointing/keeper/genesis.go
- x/checkpointing/keeper/genesis_bls.go
- x/checkpointing/keeper/genesis_test.go
- x/checkpointing/keeper/grpc_query_bls.go
- x/checkpointing/keeper/grpc_query_bls_test.go
- x/checkpointing/keeper/grpc_query_checkpoint.go
- x/checkpointing/keeper/grpc_query_checkpoint_test.go
- x/checkpointing/keeper/hooks.go
- x/checkpointing/keeper/keeper.go
- x/checkpointing/keeper/keeper_test.go
- x/checkpointing/keeper/msg_server.go
- x/checkpointing/keeper/msg_server_gas_test.go
- x/checkpointing/keeper/msg_server_test.go
- x/checkpointing/keeper/registration_state.go
- x/checkpointing/keeper/val_bls_set.go
- x/checkpointing/keeper/val_bls_set_test.go
- x/checkpointing/module.go
- x/checkpointing/prepare/proposal.go
- x/checkpointing/prepare/proposal_expected_keeper.go
- x/checkpointing/prepare/proposal_test.go
- x/checkpointing/prepare/transactions.go
- x/checkpointing/prepare/transactions_test.go
- x/checkpointing/types/bls_key.go

- x/checkpointing/types/bls_key.pb.go
- x/checkpointing/types/bls_key_test.go
- x/checkpointing/types/checkpoint.go
- x/checkpointing/types/checkpoint.pb.go
- x/checkpointing/types/checkpoint_test.go
- x/checkpointing/types/codec.go
- x/checkpointing/types/errors.go
- x/checkpointing/types/events.go
- x/checkpointing/types/events.pb.go
- x/checkpointing/types/expected_keepers.go
- x/checkpointing/types/genesis.go
- x/checkpointing/types/genesis.pb.go
- x/checkpointing/types/genesis_test.go
- x/checkpointing/types/hooks.go
- x/checkpointing/types/keys.go
- x/checkpointing/types/msgs.go
- x/checkpointing/types/msgs_test.go
- x/checkpointing/types/pop.go
- x/checkpointing/types/pop_test.go
- x/checkpointing/types/querier.go
- x/checkpointing/types/query.go
- x/checkpointing/types/query.pb.go
- x/checkpointing/types/query.pb.gw.go
- x/checkpointing/types/tx.pb.go
- x/checkpointing/types/types.go
- x/checkpointing/types/types_test.go
- x/checkpointing/types/utils.go
- x/checkpointing/types/val_bls_set.go
- x/checkpointing/vote_extensions/vote_ext.go
- x/checkpointing/vote_extensions/vote_ext_test.go
- x/costaking/README.md
- x/costaking/abci.go
- x/costaking/client/cli/query.go
- x/costaking/keeper/events.go
- x/costaking/keeper/genesis.go
- x/costaking/keeper/genesis_test.go
- x/costaking/keeper/grpc_query.go
- x/costaking/keeper/hooks_finality.go
- x/costaking/keeper/hooks_incentive.go
- x/costaking/keeper/hooks_staking.go
- x/costaking/keeper/hooks_staking_test.go
- x/costaking/keeper/integration_test.go
- x/costaking/keeper/intercept_fee_collector.go
- x/costaking/keeper/intercept_fee_collector_test.go

- x/costaking/keeper/keeper.go
- x/costaking/keeper/keeper_test.go
- x/costaking/keeper/msg_server.go
- x/costaking/keeper/msg_server_test.go
- x/costaking/keeper/params.go
- x/costaking/keeper/reward_tracker.go
- x/costaking/keeper/reward_tracker_test.go
- x/costaking/keeper/rewards_store.go
- x/costaking/keeper/rewards_store_test.go
- x/costaking/keeper/score.go
- x/costaking/module.go
- x/costaking/types/codec.go
- x/costaking/types/costaking.pb.go
- x/costaking/types/errors.go
- x/costaking/types/events.go
- x/costaking/types/events.pb.go
- x/costaking/types/expected_keepers.go
- x/costaking/types/genesis.go
- x/costaking/types/genesis.pb.go
- x/costaking/types/genesis_test.go
- x/costaking/types/keys.go
- x/costaking/types/mocked_keepers.go
- x/costaking/types/params.go
- x/costaking/types/params_test.go
- x/costaking/types/query.pb.go
- x/costaking/types/query.pb.gw.go
- x/costaking/types/rewards.go
- x/costaking/types/rewards.pb.go
- x/costaking/types/score.go
- x/costaking/types/score_test.go
- x/costaking/types/staking_cache.go
- x/costaking/types/staking_cache_test.go
- x/costaking/types/tx.pb.go
- x/epoching/README.md
- x/epoching/abci.go
- x/epoching/client/cli/README.md
- x/epoching/client/cli/query.go
- x/epoching/client/cli/tx.go
- x/epoching/genesis.go
- x/epoching/genesis_test.go
- x/epoching/keeper/delegation_pool.go
- x/epoching/keeper/delegation_pool_test.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/epochs_test.go
- x/epoching/keeper/genesis.go
- x/epoching/keeper/genesis_test.go
- x/epoching/keeper/grpc_query.go
- x/epoching/keeper/grpc_query_test.go
- x/epoching/keeper/hooks.go
- x/epoching/keeper/keeper.go
- x/epoching/keeper/keeper_test.go
- x/epoching/keeper/lifecycle_delegation.go
- x/epoching/keeper/lifecycle_validator.go
- x/epoching/keeper/migrations.go
- x/epoching/keeper/modified_staking.go
- x/epoching/keeper/msg_server.go
- x/epoching/keeper/msg_server_gas_test.go
- x/epoching/keeper/msg_server_test.go
- x/epoching/keeper/params.go
- x/epoching/keeper/params_test.go
- x/epoching/keeper/staking_functions.go
- x/epoching/migrations/v1/types.go
- x/epoching/migrations/v2/store.go
- x/epoching/migrations/v2/store_test.go
- x/epoching/module.go
- x/epoching/types/codec.go
- x/epoching/types/epoching.go
- x/epoching/types/epoching.pb.go
- x/epoching/types/epoching_test.go
- x/epoching/types/errors.go
- x/epoching/types/events.go
- x/epoching/types/events.pb.go
- x/epoching/types/expected_keepers.go
- x/epoching/types/genesis.go
- x/epoching/types/genesis.pb.go
- x/epoching/types/genesis_test.go
- x/epoching/types/hooks.go
- x/epoching/types/keys.go
- x/epoching/types/msg.go
- x/epoching/types/msg_test.go
- x/epoching/types/params.go
- x/epoching/types/params.pb.go
- x/epoching/types/params_test.go
- x/epoching/types/query.go
- x/epoching/types/query.pb.go

- x/epoching/types/query.pb.gw.go
- x/epoching/types/query_test.go
- x/epoching/types/tx.pb.go
- x/epoching/types/types.go
- x/epoching/types/validator.go
- x/epoching/types/validator_test.go
- x/finality/README.md
- 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/genesis_test.go
- x/finality/keeper/evidence.go
- x/finality/keeper/genesis.go
- x/finality/keeper/genesis_test.go
- x/finality/keeper/gov.go
- x/finality/keeper/gov_test.go
- x/finality/keeper/grpc_query.go
- x/finality/keeper/grpc_query_test.go
- x/finality/keeper/indexed_blocks.go
- x/finality/keeper/keeper.go
- x/finality/keeper/liveness.go
- x/finality/keeper/liveness_test.go
- x/finality/keeper/migrations.go
- x/finality/keeper/msg_server.go
- x/finality/keeper/msg_server_test.go
- x/finality/keeper/params.go
- x/finality/keeper/params_test.go
- x/finality/keeper/power_dist_change.go
- x/finality/keeper/power_dist_change_test.go
- x/finality/keeper/power_table.go
- x/finality/keeper/power_table_test.go
- x/finality/keeper/public_randomness.go
- x/finality/keeper/public_randomness_test.go
- x/finality/keeper/query.go
- x/finality/keeper/query_params.go
- x/finality/keeper/query_params_test.go
- x/finality/keeper/rewarding.go
- x/finality/keeper/rewarding_test.go
- x/finality/keeper/signing_info.go
- x/finality/keeper/tallying.go
- x/finality/keeper/tallying_bench_test.go
- x/finality/keeper/tallying_test.go

- x/finality/keeper/votes.go
- x/finality/keeper/votes_bench_test.go
- x/finality/migrations/v2/store.go
- x/finality/migrations/v2/store_test.go
- x/finality/module.go
- x/finality/types/codec.go
- x/finality/types/constants.go
- x/finality/types/errors.go
- x/finality/types/events.go
- x/finality/types/events.pb.go
- x/finality/types/expected_keepers.go
- x/finality/types/finality.go
- x/finality/types/finality.pb.go
- x/finality/types/finality_test.go
- x/finality/types/genesis.go
- x/finality/types/genesis.pb.go
- x/finality/types/genesis_test.go
- x/finality/types/hooks.go
- x/finality/types/keys.go
- x/finality/types/metrics.go
- x/finality/types/mocked_hooks.go
- x/finality/types/mocked_keepers.go
- x/finality/types/msg.go
- x/finality/types/msg_test.go
- x/finality/types/params.go
- x/finality/types/params.pb.go
- x/finality/types/power_table.go
- x/finality/types/power_table_test.go
- x/finality/types/query.go
- x/finality/types/query.pb.go
- x/finality/types/query.pb.gw.go
- x/finality/types/signing_info.go
- x/finality/types/signing_info_test.go
- x/finality/types/tx.pb.go
- x/finality/types/types.go
- x/incentive/README.md
- 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/genesis_test.go
- x/incentive/keeper/btc_staking_gauge.go
- x/incentive/keeper/btc_staking_gauge_test.go

- x/incentive/keeper/finality_hooks.go
- x/incentive/keeper/genesis.go
- x/incentive/keeper/genesis_test.go
- x/incentive/keeper/grpc_query.go
- x/incentive/keeper/grpc_query_test.go
- x/incentive/keeper/intercept_fee_collector.go
- x/incentive/keeper/intercept_fee_collector_test.go
- x/incentive/keeper/keeper.go
- x/incentive/keeper/keeper_test.go
- x/incentive/keeper/migrations.go
- x/incentive/keeper/msg_server.go
- x/incentive/keeper/msg_server_test.go
- x/incentive/keeper/params.go
- x/incentive/keeper/params_test.go
- x/incentive/keeper/query_delegator.go
- x/incentive/keeper/query_delegator_test.go
- x/incentive/keeper/query_params.go
- x/incentive/keeper/query_params_test.go
- x/incentive/keeper/refund_tx_decorator.go
- x/incentive/keeper/refund_tx_decorator.go
- x/incentive/keeper/refundable_msg_index.go
- x/incentive/keeper/reward_gauge.go
- x/incentive/keeper/reward_tracker.go
- x/incentive/keeper/reward_tracker_events.go
- x/incentive/keeper/reward_tracker_events_test.go
- x/incentive/keeper/reward_tracker_store.go
- x/incentive/keeper/reward_tracker_store_test.go
- x/incentive/keeper/reward_tracker_test.go
- x/incentive/keeper/store.go
- x/incentive/migrations/v2/store.go
- x/incentive/module.go
- x/incentive/types/codec.go
- x/incentive/types/errors.go
- x/incentive/types/events.go
- x/incentive/types/events.pb.go
- x/incentive/types/expected_keepers.go
- x/incentive/types/genesis.go
- x/incentive/types/genesis.pb.go
- x/incentive/types/genesis_test.go
- x/incentive/types/hooks.go
- x/incentive/types/incentive.go
- x/incentive/types/incentive.pb.go
- x/incentive/types/incentive_test.go
- x/incentive/types/keys.go

- x/incentive/types/mocked_hooks.go
- x/incentive/types/mocked_keepers.go
- x/incentive/types/msg.go
- x/incentive/types/params.go
- x/incentive/types/params.pb.go
- x/incentive/types/query.pb.go
- x/incentive/types/query.pb.gw.go
- x/incentive/types/rewards.go
- x/incentive/types/rewards.pb.go
- x/incentive/types/rewards_test.go
- x/incentive/types/tx.pb.go
- x/incentive/types/types.go
- x/mint/README.md
- x/mint/abci.go
- x/mint/abci_test.go
- x/mint/client/cli/query.go
- x/mint/keeper/genesis.go
- x/mint/keeper/genesis_test.go
- x/mint/keeper/grpc_query.go
- x/mint/keeper/grpc_query_test.go
- x/mint/keeper/keeper.go
- x/mint/module.go
- x/mint/module_test.go
- x/mint/simulation/decoder.go
- x/mint/simulation/decoder_test.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/genesis.pb.go
- x/mint/types/keys.go
- x/mint/types/mint.pb.go
- x/mint/types/minter.go
- x/mint/types/minter_test.go
- x/mint/types/query.pb.go
- x/mint/types/query.pb.gw.go
- x/monitor/client/cli/query.go
- x/monitor/client/cli/tx.go
- x/monitor/genesis.go
- x/monitor/genesis_test.go
- x/monitor/keeper/genesis.go
- x/monitor/keeper/genesis_test.go
- x/monitor/keeper/grpc_query.go
- x/monitor/keeper/grpc_query_test.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/genesis.pb.go
- x/monitor/types/genesis_test.go
- x/monitor/types/keys.go
- x/monitor/types/query.pb.go
- x/monitor/types/query.pb.gw.go

Out-of-Scope: Third party dependencies and economic attacks.

REMEDIATION COMMIT ID:

- 01ddad4
- 04aa678
- d402876
- 0bb1d14
- 50a6f97

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	2	7

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
MISSING UNBONDING/REDELEGATION ENTRY LIMIT CHECKS IN QUEUED EPOCHING MESSAGES	LOW	SOLVED - 10/29/2025

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
MSGWRAPPEDCREATEVALIDATOR CREATES UNTRACKABLE QUEUED MESSAGES DUE TO MISSING TXID AND MSGID FIELDS	LOW	SOLVED - 10/29/2025
UNBOUNDED COSTAKER ITERATION IN MSGUPDATEPARAMS ENABLES DOS	INFORMATIONAL	ACKNOWLEDGED - 10/29/2025
PROOF OF POSSESSION DOES NOT USE DOMAIN SEPARATOR FOR THE SIGNATURE	INFORMATIONAL	ACKNOWLEDGED - 10/28/2025
JAILED FINALITY PROVIDERS CAN RECEIVE DELEGATIONS	INFORMATIONAL	ACKNOWLEDGED - 10/28/2025
DELETED FINALITY PROVIDERS CAN RECEIVE DELEGATIONS	INFORMATIONAL	ACKNOWLEDGED - 10/28/2025
MISLEADING INTERNAL KEY HANDLING IN DERIVETAPROOTPKSCRIPT	INFORMATIONAL	SOLVED - 10/26/2025
DUPLICATE STAKING OUTPUTS SLIP PAST VALIDATION	INFORMATIONAL	SOLVED - 10/26/2025
DELEGATION TRANSACTION OP_RETURN IS NOT VERIFIED	INFORMATIONAL	SOLVED - 10/26/2025

7. FINDINGS & TECH DETAILS

7.1 MISSING UNBONDING/REDELEGATION ENTRY LIMIT CHECKS IN QUEUED EPOCHING MESSAGES

// LOW

Description

The `MsgWrappedUndelegate` and `MsgWrappedBeginRedelegate` in the `x/epoching` module enqueue staking operations to be executed later at the epoch boundary. These messages perform basic checks such as minimum amount validation and gas deduction upfront before being queued. However, they do not validate whether the delegator has already reached the maximum allowed number of concurrent `unbonding` or `redelegation` entries (typically 7 per delegator–validator pair). As a result, transactions that exceed this limit are accepted and enqueued successfully but will fail during epoch execution, after gas has already been consumed. This leads to unnecessary gas loss for users.

x/epoching/keeper/msg_server.go

```
233 func (ms msgServer) WrappedUndelegate(goCtx context.Context, msg *types.MsgWrappedUndelegate) (*types.Ms
234     ctx := sdk.UnwrapSDKContext(goCtx)
235     if msg.Msg == nil {
236         return nil, types.ErrNoWrappedMsg
237     }
238 
239     // verification rules ported from staking module
240     valAddr, err := sdk.ValAddressFromBech32(msg.Msg.ValidatorAddress)
241     if err != nil {
242         return nil, err
243     }
244     delegatorAddress, err := sdk.AccAddressFromBech32(msg.Msg.DelegatorAddress)
245     if err != nil {
246         return nil, err
247     }
248     if _, err := ms.stk.ValidateUnbondAmount(ctx, delegatorAddress, valAddr, msg.Msg.Amount.Amount); err
249         return nil, err
250     }
251     bondDenom, err := ms.stk.BondDenom(ctx)
252     if err != nil {
253         return nil, err
254     }
255     if msg.Msg.Amount.Denom != bondDenom {
256         return nil, errorsmod.Wrapf(
257             sdkerrors.ErrInvalidRequest, "invalid coin denomination: got %s, expected %s", msg.Msg.Amoun
258         )
259     }
260 
261     params := ms.GetParams(ctx)
262     // check if the undelegation amount is above the minimum required amount
263     if msg.Msg.Amount.Amount.LT(math.NewIntFromUint64(params.MinAmount)) {
264         return nil, errorsmod.Wrapf(
265             sdkerrors.ErrInvalidRequest,
266             "undelegation amount %s is below minimum required amount %d",
267             msg.Msg.Amount.Amount.String(),
268             params.MinAmount,
269         )
270     }
271 
272     blockHeight := uint64(ctx.HeaderInfo().Height)
273 
```

```
    if blockHeight == 0 {
        return nil, types.ErrZeroEpochMsg
    }
    blockTime := ctx.HeaderInfo().Time

    txid := tmhash.Sum(ctx.TxBytes())
    queuedMsg, err := types.NewQueuedMessage(blockHeight, blockTime, txid, msg)
    if err != nil {
        return nil, err
    }

    ms.EnqueueMsg(ctx, queuedMsg)

    // charge gas for executing the message later
    ctx.GasMeter().ConsumeGas(params.ExecuteGas.Undelegate, "epoching undelegate enqueue fee")

    err = ctx.EventManager().EmitTypedEvents(
        &types.EventWrappedUndelegate{
            DelegatorAddress: msg.Msg.DelegatorAddress,
            ValidatorAddress: msg.Msg.ValidatorAddress,
            Amount:           msg.Msg.Amount.Amount.Uint64(),
            Denom:            msg.Msg.Amount.GetDenom(),
            EpochBoundary:    ms.GetEpoch(ctx).GetLastBlockHeight(),
        },
    )
    if err != nil {
        return nil, err
    }

    return &types.MsgWrappedUndelegateResponse{}, nil
}

func (ms msgServer) WrappedBeginRedelegate(goCtx context.Context, msg *types.MsgWrappedBeginRedelegate)
    ctx := sdk.UnwrapSDKContext(goCtx)
    if msg.Msg == nil {
        return nil, types.ErrNoWrappedMsg
    }

    // verification rules ported from staking module
    valSrcAddr, err := sdk.ValAddressFromBech32(msg.Msg.ValidatorSrcAddress)
    if err != nil {
        return nil, err
    }
    delegatorAddress, err := sdk.AccAddressFromBech32(msg.Msg.DelegatorAddress)
    if err != nil {
        return nil, err
    }
    if _, err := ms.stk.ValidateUnbondAmount(ctx, delegatorAddress, valSrcAddr, msg.Msg.Amount.Amount);
        return nil, err
    }
    bondDenom, err := ms.stk.BondDenom(ctx)
    if err != nil {
        return nil, err
    }
    if msg.Msg.Amount.Denom != bondDenom {
        return nil, errorsmod.Wrapf(
            sdkerrors.ErrInvalidRequest, "invalid coin denomination: got %s, expected %s", msg.Msg.Amoun
        )
    }

    params := ms.GetParams(ctx)
    // check if the redelegation amount is above the minimum required amount
    if msg.Msg.Amount.Amount.LT(math.NewIntFromUint64(params.MinAmount)) {
        return nil, errorsmod.Wrapf(
            sdkerrors.ErrInvalidRequest,
            "redelegation amount %s is below minimum required amount %d",
            msg.Msg.Amount.Amount.String(),
            params.MinAmount,
        )
    }

    if _, err := sdk.ValAddressFromBech32(msg.Msg.ValidatorDstAddress); err != nil {
        return nil, err
```

```
    }

    blockHeight := uint64(ctx.HeaderInfo().Height)
    if blockHeight == 0 {
        return nil, types.ErrZeroEpochMsg
    }
    blockTime := ctx.HeaderInfo().Time

    txid := tmhash.Sum(ctx.TxBytes())
    queuedMsg, err := types.NewQueuedMessage(blockHeight, blockTime, txid, msg)
    if err != nil {
        return nil, err
    }

    ms.EnqueueMsg(ctx, queuedMsg)

    // charge gas for executing the message later
    ctx.GasMeter().ConsumeGas(params.ExecuteGas.BeginRedelegate, "epoching Redelegate enqueue fee")

    err = ctx.EventManager().EmitTypedEvents(
        &types.EventWrappedBeginRedelegate{
            DelegatorAddress:            msg.Msg.DelegatorAddress,
            SourceValidatorAddress:      msg.Msg.ValidatorSrcAddress,
            DestinationValidatorAddress: msg.Msg.ValidatorDstAddress,
            Amount:                      msg.Msg.Amount.Amount.Uint64(),
            Denom:                       msg.Msg.Amount.GetDenom(),
            EpochBoundary:               ms.GetEpoch(ctx).GetLastBlockHeight(),
        },
    )
    if err != nil {
        return nil, err
    }

    return &types.MsgWrappedBeginRedelegateResponse{}, nil
}
```

BVSS

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

Recommendation

It is recommended to utilize the `HasMaxUnbondingDelegationEntries` and `HasMaxRedelegationEntries` helpers before enqueuing the message to ensure entry limits are not exceeded and prevent wasted gas on failing transactions.

Remediation Comment

SOLVED: The **Babylon Labs team** resolved the finding by validating unbonding and redelagtion entry limits before enqueuing the message.

Remediation Hash

`01ddad4b899af818a1e23cb652a4ed59eaa8e1f9`

7.2 MSGWRAPPEDCREATEVALIDATOR CREATES UNTRACKABLE QUEUED MESSAGES DUE TO MISSING TXID AND MSGID FIELDS

// LOW

Description

The `MsgWrappedCreateValidator` handler in the `x/checkpointing` module allows validator registration by enqueuing a `MsgCreateValidator` into the `x/epoching` module for execution at the epoch boundary. This involves validating the request, storing the BLS key, locking the delegation funds, and constructing a `QueuedMessage` for later processing.

However, the handler manually constructs this `QueuedMessage` without setting the required `TxId` and `MsgId` fields. As a result, the message is queued with `nil` identifiers, making it untraceable in logs and emitted events during `Epoching` module `EndBlocker` execution. This breaks visibility into validator creation activity, debugging and monitoring of failed validator registrations.

x/checkpointing/keeper/msg_server.go

```
28  func (m msgServer) WrappedCreateValidator(goCtx context.Context, msg *types.MsgWrappedCreateValidator) (
29      ctx := sdk.UnwrapSDKContext(goCtx)
30
31      // stateless checks on the inside MsgCreateValidator msg
32      executeGas, err := m.k.epochingKeeper.CheckMsgCreateValidator(ctx, msg.MsgCreateValidator)
33      if err != nil {
34          return nil, err
35      }
36
37      valAddr, err := sdk.ValAddressFromBech32(msg.MsgCreateValidator.ValidatorAddress)
38      if err != nil {
39          return nil, err
40      }
41
42      // store BLS public key
43      err = m.k.CreateRegistration(ctx, *msg.Key.Pubkey, valAddr)
44      if err != nil {
45          return nil, err
46      }
47
48      if ctx.HeaderInfo().Height == 0 {
49          // no need to put in a queue if it is a genesis transactions
50          err = m.k.epochingKeeper.StkMsgCreateValidator(ctx, msg.MsgCreateValidator)
51          if err != nil {
52              return nil, err
53          }
54          return &types.MsgWrappedCreateValidatorResponse{}, nil
55      }
56
57      // enqueue the msg into the epoching module
58      queueMsg := epochingtypes.QueuedMessage{
59          Msg: &epochingtypes.QueuedMessage_MsgCreateValidator{MsgCreateValidator: msg.MsgCreateValidator}
60      }
61      // lock the delegation amount to ensure funds are available when the queued message executes
62      // this prevents spam attacks by requiring actual fund ownership and guarantees successful execution
63      err = m.k.epochingKeeper.LockFundsForDelegateMsgs(ctx, &queueMsg)
64      if err != nil {
65          return nil, err
66      }
67
```

```
68      m.k.epochingKeeper.EnqueueMsg(ctx, queueMsg)
69
70      // charge gas upfront for executing the message later at epoch end
71      ctx.GasMeter().ConsumeGas(executeGas, "epoching create validator execution fee")
72
73      return &types.MsgWrappedCreateValidatorResponse{}, nil
74  }
```

BVSS

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

Recommendation

It is recommended to construct the `QueuedMessage` using the standard helper `types.NewQueuedMessage(blockHeight, blockTime, txId, msg)` as used in other wrapped message handlers to ensure the `TxId` and `MsgId` fields are correctly populated.

Remediation Comment

SOLVED: The **Babylon Labs team** resolved the finding by constructing the `QueuedMessage` using the standard helper.

Remediation Hash

04aa6782e14622a6f9771bccd907a662fab17f95

7.3 UNBOUNDED COSTAKER ITERATION IN MSGUPDATEPARAMS ENABLES DOS

// INFORMATIONAL

Description

The `MsgUpdateParams` message is executed via governance on the Babylon chain to update `x/costaking` module parameters. During proposal execution, the handler checks whether the new `ScoreRatioBtcByBaby` differs from the current value; if so, it invokes `UpdateAllCostakersScore`, which iterates over all `CostakerRewardsTracker` entries in state. For each, it recomputes the costaker's score, distributes any pending rewards to the incentive gauge, updates the tracker, and finally recalculates the global `CurrentRewards.TotalScore`.

However, this iteration is unbounded and unbatched, meaning both gas usage and execution time scale linearly with the number of costakers. If the number is too large, the transaction may consistently exceed block gas limits or ABCI timeouts, causing governance proposal execution to fail. This results in a denial of service, preventing parameter changes like reward formula updates from ever being applied without a chain upgrade or manual intervention.

x/costaking/keeper/msg_server.go

```
12  func (k Keeper) UpdateAllCostakersScore(ctx context.Context, scoreRatioBtcByBaby math.Int) error {
13      totalScore := math.ZeroInt()
14
15      endedPeriod, err := k.IncrementRewardsPeriod(ctx)
16      if err != nil {
17          return err
18      }
19
20      currentRwd, err := k.GetCurrentRewards(ctx)
21      if err != nil {
22          return err
23      }
24
25      err = k.IterateCostakers(ctx, func(costaker sdk.AccAddress, rwdTracker types.CostakerRewardsTracker)
26          deltaScoreChange := rwdTracker.UpdateScore(scoreRatioBtcByBaby)
27
28          totalScore = totalScore.Add(rwdTracker.TotalScore)
29          if deltaScoreChange.IsZero() {
30              // if there is no change from previous score, continue
31              return k.setCostakerRewardsTracker(ctx, costaker, rwdTracker)
32          }
33
34          if err := k.CalculateCostakerRewardsAndSendToGauge(ctx, costaker, endedPeriod); err != nil {
35              return err
36          }
37
38          if err := k.setCostakerRewardsTracker(ctx, costaker, rwdTracker); err != nil {
39              return err
40          }
41
42          return k.initializeCoStakerRwdTracker(ctx, costaker)
43      })
44      if err != nil {
45          return err
46      }
47
48
```

```
    currentRwd.TotalScore = totalScore
    if err := currentRwd.Validate(); err != nil {
        return err
    }

    return k.UpdateCurrentRewardsTotalScore(ctx, totalScore)
}

// IterateCostakers iterates over all the costakers.
func (k Keeper) IterateCostakers(ctx context.Context, it func(addr sdk.AccAddress, rwdTracker types.Cost
    return k.costakerRewardsTracker.Walk(ctx, nil, func(key []byte, value types.CostakerRewardsTracker)
        addr := sdk.AccAddress(key)
        err = it(addr, value)
        if err != nil {
            return true, err
        }
        return false, nil
    })
}
```

Proof of Concept

```
// Place the test in  x/costaking/keeper/msg_server_update_params_test.go file

func TestMsgUpdateParamsUpdateAllCostakersScore_HeavyLoad(t *testing.T) {
    ctrl := gomock.NewController(t)
    t.Cleanup(ctrl.Finish)

    ictvK := types.NewMockIncentiveKeeper(ctrl)
    k, ctx := NewKeeperWithMockIncentiveKeeper(t, ictvK)

    const numCostakers = 4000000
    initialRatio := sdkmath.NewInt(50)
    newRatio := sdkmath.NewInt(10)

    // Set initial parameters
    dp := types.DefaultParams()
    dp.ScoreRatioBtcByBaby = initialRatio
    require.NoError(t, k.SetParams(ctx, dp))

    // Generate costakers with varying scores
    costakers := make([]sdk.AccAddress, 0, numCostakers)
    for i := 0; i < numCostakers; i++ {
        addr := datagen.GenRandomAddress()
        costakers = append(costakers, addr)

        btcAmt := sdkmath.NewInt(int64(1000 + (i % 1000))) // 1000-1999 sats
        babyAmt := btcAmt.Mul(initialRatio)                // ensure non-zero proportional score
        err := k.costakerModifiedActiveAmounts(ctx, addr, btcAmt, babyAmt)
        require.NoError(t, err)
    }

    currentRwd, err := k.GetCurrentRewards(ctx)
    require.NoError(t, err)
    t.Logf("Initial total score: %s", currentRwd.TotalScore)

    // Deposit rewards and finalize period
    totalDeposited := sdk.NewCoins(sdk.NewCoin(appparams.DefaultBondDenom, sdkmath.NewInt(100_000_000)))
    require.NoError(t, k.AddRewardsForCostakers(ctx, totalDeposited))

    t.Logf("=== STARTING GOVERNANCE PARAMETER UPDATE for %d costakers ===", numCostakers)
    t.Logf("ScoreRatioBtcByBaby: %s -> %s", initialRatio, newRatio)

    start := time.Now()
    err = k.UpdateAllCostakersScore(ctx, newRatio)
    elapsed := time.Since(start)

    require.NoError(t, err)
```

```
    t.Logf("Execution time for UpdateAllCostakersScore with %d costakers: %s", numCostakers, elapsed)

    currentRwd, err = k.GetCurrentRewards(ctx)
    require.NoError(t, err)
    t.Logf("Final total score: %s", currentRwd.TotalScore)
    t.Logf("Total time for governance update over %d costakers: %v", numCostakers, elapsed)
}
```

```
msg_server_update_params_test.go:188: === STARTING GOVERNANCE PARAMETER UPDATE for 4000000 costakers ===
msg_server_update_params_test.go:189: ScoreRatioBtcByBaby: 50 -> 10
msg_server_update_params_test.go:196: Execution time for UpdateAllCostakersScore with 4000000 costakers: 21.8204715s
msg_server_update_params_test.go:200: Final total score: 5998000000
msg_server_update_params_test.go:201: Total time for governance update over 4000000 costakers: 21.8204715s
```

BVSS

AO:A/AC:L/AX:H/R:P/S:U/C:N/A:H/I:N/D:N/Y:N (1.2)

Recommendation

To address this finding, It is recommended to process costaker score recalculations in fixed size batches, rather than iterating over the entire set in a single governance-triggered transaction.

Remediation Comment

ACKNOWLEDGED: The **Babylon Labs team** acknowledged the finding and stated the following:

> If there is a change in the `ScoreRatioBtcByBaby` value, it needs to update all the costakers score in the same block. Otherwise some costakers might have one ratio while a portion of other costakers will have another ratio with an different amount and diverging the amount of rewards that each costaker is entitled to have.

7.4 PROOF OF POSSESSION DOES NOT USE DOMAIN SEPARATOR FOR THE SIGNATURE

// INFORMATIONAL

Description

It is recommended to sign to Proof of possession (PoP) using a domain separators (chain id, version, module, address) so that the signature cannot be reused in external protocols.

The attack scenario is as follows:

1. The victim signs its babylon address using its BTC secret key.
2. The attacker sees the signature in the babylon transaction explorer and tries to find other protocols that require a signature from a BTC key and show knowledge of the signed payload.

BVSS

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

Recommendation

To address this finding, it is recommended to use domain separator for signature.

Remediation Comment

ACKNOWLEDGED: The **Babylon Labs team** acknowledged the finding.

7.5 JAILED FINALITY PROVIDERS CAN RECEIVE DELEGATIONS

// INFORMATIONAL

Description

The Jailing mechanism happens when a finality provider misses too many votes in the signed-blocks window. Jailing sets `Jailed = true`, removes voting power, and requires an explicit unjail after the `JailDuration`. During the assessment, It was found that delegations were still possible in Babylon on jailed finality providers. There was a single check that the FP was not slashed, but nothing on jailed ones.

In `btc_delegations.go`: there are verifications that the finality provider is not slashed, but nothing about jailed finality providers:

```
507 func (k Keeper) validateStakedFPs(ctx sdk.Context, fpBTCPKs []bbn.BIP340PubKey) error {
508     for i := range fpBTCPKs {
509         fpBTCPK := fpBTCPKs[i]
510         fp, err := k.GetFinalityProvider(ctx, fpBTCPK)
511         if err != nil {
512             return types.ErrFpNotFound.Wrapf("finality provider pk %s is not found: %v", fpBTCPK.Marshal
513         }
514         if fp.IsSlashed() {
515             return types.ErrFpAlreadySlashed.Wrapf("finality key: %s", fpBTCPK.MarshalHex())
516         }
517     }
518     return nil
519 }
```

NOTE: This behaviour is intended to give the stakers the ability to unbond.

BVSS

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

Recommendation

To address this finding, it is recommended to prevent jailed finality providers to receive delegations.

Remediation Comment

ACKNOWLEDGED: The **Babylon Labs team** acknowledged the finding.

7.6 DELETED FINALITY PROVIDERS CAN RECEIVE DELEGATIONS

// INFORMATIONAL

Description

The `CreateBTCDelegation` only rejects unknown or slashed finality providers; it never asks whether the target FP was soft-deleted. FPs can be soft-deleted during upgrade handler or or genesis import, during which the network permanently flips finalityProvidersDeleted for that BTC key. `CreateBTCDelegation` still accepts staking to that key because `validateStakedFPs` only checks existence and `IsSlashed()`. The delegator's BTC is locked in the staking script and the delegation proceeds through covenant signing as usual.

Once live, finality refuses every vote from that FP, so the provider always has zero voting power: the delegation never counts toward quorum and earns no finality rewards. The stake is effectively stranded until the user unbonds, wasting capital for the full bonding/unbonding window. Because soft-deleted keys come from real migrations (removing duplicate Babylon addresses) and can keep advertising deposit info, any delegator following old instructions can hit this today.

In `btc_delegations.go`, there is no verification that the FP was not soft deleted;

```
507 | func (k Keeper) validateStakedFPs(ctx sdk.Context, fpBTCPKs []bbn.BIP340PubKey) error {
508 |     for i := range fpBTCPKs {
509 |         fpBTCPK := fpBTCPKs[i]
510 |         fp, err := k.GetFinalityProvider(ctx, fpBTCPK)
511 |         if err != nil {
512 |             return types.ErrFpNotFound.Wrapf("finality provider pk %s is not found: %v", fpBTCPK.Marshal
513 |         }
514 |         if fp.IsSlashed() {
515 |             return types.ErrFpAlreadySlashed.Wrapf("finality key: %s", fpBTCPK.MarshalHex())
516 |         }
517 |     }
518 |     return nil
519 | }
```

For reference FPs are banned from voting, in `msg_server.go`:

```
80 | if ms.BTCStakingKeeper.IsFinalityProviderDeleted(ctx, req.FpBtcPk) {
81 |     return nil, types.ErrFinalityProviderIsDeleted.Wrapf("fp_btc_pk_hex: %s", req.FpBtcPk.MarshalHex())
82 | }
```

NOTE: this behaviour is intended to give the staker ability to unbond

BVSS

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

Recommendation

To address this finding, it is recommended to prevent deleted finality providers to receive delegations.

Remediation Comment

ACKNOWLEDGED: The **Babylon Labs team** acknowledged the finding as an intended behaviour to give stakers the ability to unbond from Jailed Finality Providers.

7.7 MISLEADING INTERNAL KEY HANDLING IN DERIVETAPROOTPKSCRIPT

// INFORMATIONAL

Description

The helper ignores its `internalPubKey` argument and always uses `unspendableKeyPathKey`, so any future caller expecting to supply a different Taproot internal key would silently produce the wrong script. Clarifying or fixing this behavior (either using the parameter or removing it) will prevent configuration drift or misuse.

In `btcstaking/types.go`, the unspendable key `internalPubKey` is passed as argument but unused. Instead, the address is derived from the package-level declared `unspendableKeyPathKey`:

```
84  func DeriveTaprootPkScript(
85      tapScriptTree *txscript.IndexedTapScriptTree,
86      internalPubKey *btcec.PublicKey,
87      net *chaincfg.Params,
88  ) ([]byte, error) {
89      taprootAddress, err := DeriveTaprootAddress(
90          tapScriptTree,
91          &unspendableKeyPathKey,
92          net,
93      )
94
95      if err != nil {
96          return nil, err
97      }
98
99      taprootPkScript, err := txscript.PayToAddrScript(taprootAddress)
100
101     if err != nil {
102         return nil, err
103     }
104
105     return taprootPkScript, nil
106 }
```

BVSS

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

Recommendation

To address this issue, it is recommended to update `DeriveTaprootPkScript` to use the provided `internalPubKey` argument instead of the hardcoded `unspendableKeyPathKey`.

Remediation Comment

SOLVED: The **Babylon Labs team** resolved the finding by removing the `internalPubKey` parameter from `DeriveTaprootPkScript`.

Remediation Hash

d4028766c46a2e3b2be00befb9c94c6d5902c186

7.8 DUPLICATE STAKING OUTPUTS SLIP PAST VALIDATION

// INFORMATIONAL

Description

During `CreateBTCDelegation`, the `GetOutputIdxInBTCTx` function stops at the first transaction output whose script and amount match the expected staking output. If a delegator crafts a staking transaction that repeats that exact output, the helper still returns success, leaving other identical outputs unreported. Downstream logic assumes uniqueness and may treat the wrong output as the delegation. Unlike `babylon/btcstaking/identifiable_staking.go`, which aborts when it sees multiple matches, this helper silently ignores extras.

Babylon can accept a malformed or malicious staking transaction that contains multiple identical staking outputs. That ambiguity makes it hard to track the real bonding source and opens the door for replay or slash/accounting bugs when the “wrong” output is later referenced.

In `babylon/types/btcutils.go`:

```
89  func GetOutputIdxInBTCTx(tx *wire.MsgTx, output *wire.TxOut) (uint32, error) {
90      for i, txOut := range tx.TxOut {
91          if bytes.Equal(txOut.PkScript, output.PkScript) && txOut.Value == output.Value {
92              if i <= math.MaxUint32 {
93                  return uint32(i), nil
94              } else {
95                  return 0, fmt.Errorf("output index out of acceptable range")
96              }
97          }
98      }
99
100     return 0, fmt.Errorf("output not found")
101 }
```

BVSS

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

Recommendation

To address this finding, it is recommended to reject transactions that expose more than one matching output at the point of lookup. The function should track whether a matching output has already been encountered and return an explicit error if a second match is detected.

Remediation Comment

SOLVED: The **Babylon Labs team** resolved the finding by reject transactions that expose multiple matching outputs.

Remediation Hash

0bb1d1424276e63ed74ca1da15fe5d58e83b8e1a

7.9 DELEGATION TRANSACTION OP_RETURN IS NOT VERIFIED

// INFORMATIONAL

Description

The `docs/transaction-impl-spec.md` document specifies the requirements to consider a staking transaction valid. Among these, the taproot scripts, the taproot internal key, but also instructions for the `OP_RETURN` field, which should contain: `global_parameters.tag`, `version`, `staker_pk`, `finality_provider_pk`, `staking_time`.

It was found that the `btcstaking` module did not verify that the bitcoin transaction used that `OP_RETURN` format, therefore the transactions are accepted and included in babylon delegations but cannot be monitored by the offchain services. The Slashing protection is compromised, invalid stakes might not be slashed if they're not detected. Attacker could also include `OP_RETURN` with wrong staker key, FP key, or staking time.

In `docs/transaction-impl-spec.md`, the documentation states requirements for the `OP_RETURN` field:

```
- Have a Taproot output which has the key spending path disabled
and commits to a script tree composed of three scripts::
timelock script, unbonding script, slashing script.
This output is henceforth known as the staking_output and
the value in this output is known as staking_amount
- Have OP_RETURN output which contains: global_parameters.tag,
 version, staker_pk, finality_provider_pk, staking_time
- All the values must be valid for the global_parameters which are applicable at
the height in which the staking transaction is included in the BTC ledger.
```

The fields are described as follows:

- `Tag` - 4 bytes, a tag which is used to identify the staking transaction among other transactions in the Bitcoin ledger. It is specified in the `global_parameters.Tag` field.
- `Version` - 1 byte, the current version of the `OP_RETURN` output.
- `StakerPublicKey` - 32 bytes, staker public key. The same key must be used in the scripts used to create the Taproot output in the staking transaction.
- `FinalityProviderPublicKey` - 32 bytes, finality provider public key. The same key must be used in the scripts used to create the Taproot output in the staking transaction.
- `StakingTime` - 2 bytes big-endian unsigned number, staking time. The same timelock time must be used in scripts used to create the Taproot output in the staking transaction.

BVSS

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

Recommendation

To address this issue, it is recommended to update the `btcstaking` logic to parse the `OP_RETURN` output and enforce strict validation against the format and rules outlined in the `transaction-impl-spec.md` document.

Remediation Comment

SOLVED: The **Babylon Labs team** resolved the finding by updating documentation to specify that the **OP_RETURN** output format is only applicable to phase-1 BTC staking transactions.

Remediation Hash

50a6f97d43c97c8b33cedc1358dbb6e8561c5db6
