Binary Builders Audit Report

Date of Engagement: July 17th, 2023 - Oct 17th, 2023

Document revision history

Version	Modification	Date	Author
0.1	Initial creation	2023-07-27	Onur Akpolat
0.2	Add Mint and ParamFilter audit	2023-08-09	Frojdi Dymylja
0.3	Add Tokenfilter audit	2023-08-14	Aleksandr Bezobchuk
0.4	Add Upgrade audit	2023-08-17	Aleksandr Bezobchuk
0.5	Update document structure add disclaimer	2023-08-17	Onur Akpolat
0.6	Add QGB Solidity audit	2023-09-19	Onur Akpolat
1.0	Add QGB Go audit	2023-10-17	Onur Akpolat

Contacts

Contact	Company	Email
Marko Baricevic	Binary Builders	marko@binary.builders
Aleksandr Bezobchuk	Binary Builders	bez@binary.builders
Frojdi Dymylja	Binary Builders	frojdi@binary.builders
Onur Akpolat	Binary Builders	onur@binary.builders

Executive Overview

Introduction

Celestia engaged our team of blockchain auditors to conduct a security audit on their smart contracts and modules beginning on July 15th, 2023 and ending on August 15th, 2023. The security assessment was scoped to the smart contracts and modules provided in the following Celestia GitHub repositories:

https://github.com/celestiaorg/celestia-app/tree/main/x/min
https://github.com/celestiaorg/celestia-app/tree/main/x/paramfilter
https://github.com/celestiaorg/celestia-app/tree/main/x/qgb
https://github.com/celestiaorg/quantum-gravity-bridge/tree/master/src
https://github.com/celestiaorg/celestia-app/tree/main/x/tokenfilter
https://github.com/celestiaorg/celestia-app/tree/main/x/upgrade

Audit Summary

The team was provided one month for the engagement and assigned two security engineers, Frojdi Dymylja and Aleksandr Bezobchuk, to audit the security of the smart contracts and modules. Both security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract and module functions operate as intended
- Identify potential security issues with the smart contracts and modules

In summary, our team did not detect any critical security vulnerabilities but did identify several informational issues that can be addressed by the Celestia team.

Test Approach & Methodology

Our team performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart contract and module manual code review and walkthrough
- Graphing out functionality and contract logic/connectivity/functions
- Manual testing & assessment of use and safety for the critical variables and functions in scope to identify any vulnerability classes
- Scanning of contract files for vulnerabilities, security hotspots, or bugs.

Risk Methodology

Vulnerabilities or issues observed by our team are ranked based on the risk assessment methodology by measuring the LIKELIHOOD of a security incident and the IMPACT should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - INFORMATIONAL

Scope

The security assessment was scoped to the following modules:

Module	**Repository**
Mint	https://github.com/celestiaorg/celestia-app/tree/main/x/min
Paramfilter	https://github.com/celestiaorg/celestia-app/tree/main/x/paramfilter
QGB	https://github.com/celestiaorg/celestia-app/tree/main/x/qgb https://github.com/celestiaorg/quantum-gravity-bridge/tree/master/src
Tokenfilter	https://github.com/celestiaorg/celestia-app/tree/main/x/tokenfilter
Upgrade	https://github.com/celestiaorg/celestia-app/tree/main/x/upgrade

Audit Branch/Tag: **v1.0.0-rc10**
SDK Version: **v0.46.13**
IBC Version: **v6.2.0**
CometBFT (Tendermint) Version: **v0.34.28**

Future Audits

Our team suggests the need for a new security audit whenever important new functionality is developed in the core of the protocol. Focusing on the base contract, as well as on the main modules (staking, vesting, and governance) would be critical.

Moreover, it is essential to audit the security of new upcoming projects that use the Celestia protocol, as the way they interface with the core protocol can create security vulnerabilities in third-party apps.

In summary, it would be very beneficial for the project to perform a security audit in the following scenarios:

- Changes have been made to the base contract
- Changes have been made to the staking module
- Changes have been made to the vesting module
- New modules have been developed
- New projects aiming to use the Celestia protocol

Assessment Summary & Findings Overview

- Several modules use deprecated methods and could benefit from adopting more recent approaches.
- The code could benefit from improved legibility through clearer variable naming and documentation.
- One issue identified involves reliance on the previous block's time in the token minting process, which may be a concern in future CometBFT developments.

These issues are mostly informational and have low risk. Recommendations have been provided for each finding.

Findings & Tech Details

Mint

This audit of the `x/mint` module in the `celestia-app` highlights several areas for improvement, especially concerning data storage strategies and naming conventions. Implementing the provided suggestions can improve the clarity, efficiency, and maintainability of the module.

Finding 001 - Shortening of KeyMinter

ID	001
Finding	Shortening of KeyMinter
Severity	0 - Informational
Description	The bytes key `KeyMinter` is longer than necessary. Shortening it to something like `[]byte{0x0`, will save minimal storage space and align better with standard module storage practices. Additionally, it paves the way for smoother transitions to collections.
Recommendation	Trim the length of KeyMinter.
Code References	[keys.go#L4](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/types/keys.go#L4)

Finding 002 - Shortening of KeyGenesisTime

ID	002
Finding	Shortening of KeyGenesisTime
Severity	0 - Informational
Description	Similar to `KeyMinter`, the `KeyGenesisTime` can also benefit from a reduction in length.
Recommendation	Reduce the length of KeyGenesisTime.
Code References	[keys.go#L7](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/types/keys.go#L7)

Finding 003 - Streamlined Encoding of GenesisTime

ID	003
Finding	Streamlined Encoding of GenesisTime
Severity	0 - Informational
Description	The `GenesisTime` value is currently stored in state via an intermediary protobuf representation. This intermediate step is not essential as there are simpler methods to encode time to bytes, such as `sdk.FormatTimeBytes`.
Recommendation	Transition to using direct encoding methods like `sdk.FormatTimeBytes` for a more straightforward and efficient encoding process.
Code References	[mint.proto#L37](https://github.com/celestiaorg/celestia-app/blob/main/proto/celestia/mint/v1/mint.proto#L37) [keeper.go#L63](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/keeper/keeper.go#L63) [keeper.go#L70](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/keeper/keeper.go#L70)

Finding 004 - Renaming of target inflation rate

ID	004
Finding	Renaming of target inflation rate
Severity	0 - Informational
Description	The term "target inflation rate" can be misleading. A more apt name would clearly indicate that this rate is the minimum inflation rate expected.
Recommendation	Consider renaming to a more descriptive term, such as `minimum_target_inflation_rate` or another name that better communicates its function.
Code References	[constants.go#L30](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/types/constants.go#L30)

Finding 005 - Initialization of Inflation Constants

ID	005
Finding	Initialization of Inflation Constants
Severity	0 - Informational
Description	Inflation constants are currently reallocated every time they are accessed. This frequent reallocation is not necessary and may result in minor performance overheads.
Recommendation	Initialize inflation constants as variables once, and reuse them when needed, promoting efficiency and consistency.
Code References	[constants.go#L22](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/types/constants.go#L22) [constants.go#L26](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/types/constants.go#L26) [constants.go#L30](https://github.com/celestiaorg/celestia-app/blob/main/x/mint/types/constants.go#L30)

Finding 006 - AppModule Updates

ID	006
Finding	AppModule Updates
Severity	0 - Informational
Description	There are a few interface method implementations that could use re-evaluation. Specifically: ● The legacy querier is deprecated and newer chains should avoid implementing functionality that is now deprecated (see: `LegacyQuerierHandler`). ● Double check that all types and interfaces are registered in `RegisterInterfaces`.
Recommendation	See description.
Code References	x/mint/module.go

Finding 007 - Implement gRPC Query Handlers

ID	007
Finding	Implement gRPC Query Handlers
Severity	0 - Informational
Description	Currently, queries are handled via legacy querier methods via a `sdk.Querier`. Instead, consider implementing a gRPC wrapper around a `Keeper` which implements the gRPC service and register this in `module.go`. This also avoids needless JSON marshalling.
Recommendation	See description.
Code References	x/mint/keeper/querier.go

Finding 008 - Minter Evaluation

ID	008
Finding	Minter Evaluation
Severity	0 - Informational
Description	There are a few minor bits in the minter that could minor linting and evaluation. Specifically, ● There is a needless cast to int64 in `yearsSinceGenesis`. ● `CalculateBlockProvision` should consider ensuring previous is indeed less than current time and perhaps error when that is not the case. ● Avoid needless `int64` casting in `CalculateBlockProvision` and `CalculateInflationRate`.
Recommendation	See description.
Code References	x/mint/types/minter.go

Finding 009 - Reliance on Previous Block Time

ID	009
Finding	Reliance on Previous Block Time
Severity	3 - Very low
Description	`CalculateBlockProvision` uses the previous block's time in order to calculate the number of tokens to mint each block. Since in CometBFT, the proposer, or a cabal of validators with sufficient power, of a block could manipulate the block time in order to skew or impact the result of `CalculateBlockProvision`. In practice, there has been zero evidence of this happening. However, as a defensive measure, until CometBFT implements some notion of PBTS (proposer-based timestamps), ABCI++ should be explored to evaluate timestamps and ensure their integrity.
Recommendation	Explore ABCI++ to evaluate timestamps.
Code References	x/mint/abci.go

Paramfilter

The audit of the `ParamFilter` module in the `celestia-app` revealed several areas of potential improvement, primarily focused on ensuring configuration accuracy, reducing code redundancy, and refining the data structures. Implementing the recommended changes would further enhance the quality, clarity, and maintainability of the module.

Finding 010 - Use Struct as Map Key for Blocked Params

ID	010
Finding	Use Struct as Map Key for Blocked Params
Severity	0 - Informational

Description	The current implementation uses [2]string for blocked params. It would be more idiomatic and type-safe to use a defined struct like struct{Module string, Key string}.
Recommendation	Implement the use of the recommended struct format.
Code References	[gov_handler.go#L22](https://github.com/celestiaorg/celestia-app/blob/main/x/paramfilter/gov_handler.go#L22) [gov_handler.go#L25](https://github.com/celestiaorg/celestia-app/blob/main/x/paramfilter/gov_handler.go#L25) [gov_handler.go#L32](https://github.com/celestiaorg/celestia-app/blob/main/x/paramfilter/gov_handler.go#L32) [gov_handler.go#L17](https://github.com/celestiaorg/celestia-app/blob/main/x/paramfilter/gov_handler.go#L17)

Finding 011 - Check for Duplicate Blocked Params

ID	011
Finding	Check for Duplicate Blocked Params
Severity	0 - Informational
Description	The current code does not validate or check for parameters that are blocked multiple times. Although overwriting occurs and this doesn't lead to functional issues, it might be indicative of misconfiguration.
Recommendation	Implement a check for duplicate blocked parameters to prevent potential misconfigurations
Code References	gov_handler.go#L38

Finding 012 - Verify Subspace Existence

ID	012
Finding	Verify Subspace Existence
Severity	0 - Informational
Description	The code lacks a check for the existence of the subspace when creating the governance handler.
Recommendation	Introduce a validation step that ensures blocked params are part of the subspace.
Code References	[gov_handler.go#L38](https://github.com/celestiaorg/celestia-app/blob/main/x/paramfilter/gov_handler.go#L38)

Finding 013 - Middleware Approach for Param Change Proposal

ID	013
Finding	Middleware Approach for Param Change Proposal
Severity	0 - Informational
Description	The code for the param change proposal duplicates logic from the params module. This redundancy could lead to maintenance issues if the original params module is modified.
Recommendation	Adopt a middleware approach, letting the `ParamFilter` module rely on the param handler provided by the params module. This ensures continuity and minimizes the scope of `ParamFilter` to its intended functionality.
Code References	[gov_handler.go#L62:L74](https://github.com/celestiaorg/celestia-app/blob/main/x/paramfilter/gov_handler.go#L62:L74)

Finding 014 - Gov Handler Improvements

ID	014
Finding	Gov Handler Improvements
Severity	0 - Informational
Description	There are two minor programmatic improvements that could be made to the `ParamBlockList` type and implementation that would improve legibility. • The argument '`blockedParams ...[2]string)`' provided to `NewParamBlockList` could instead be a concept type encapsulating the subspace and key as fields. This would make it more clear what the arguments are meant for. In addition, a `String` method could be defined on this type to avoid multiple error-prone calls to `fmt.Sprintf`. • The message logged in `handleParameterChangeProposal` should avoid `fmt.Sprintf` call and instead use the expected API of passing key/value pairs.
Recommendation	Implement recommended changes in the code, as detailed in the audit.
Code References	x/paramfilter/gov_handler.go

Upgrade

No Findings. Apart from tests and codec registration, there was no real code to audit in the `x/upgrade` module..

Tokenfilter

Finding 015 - Unclear Wrapping of ICS4

ID	015
Finding	Unclear Wrapping of ICS4
Severity	0 - Informational
Description	It's unclear why there needs to be a wrapping of the `porttypes.ICS4Wrapper` via a `Keeper` type? Consider adding additional documentation on why the type needs to be wrapped by a proprietary `Keeper` or consider removing it altogether.
Recommendation	See description.
Code References	

Finding 016 - TokenFilterMiddleware Documentation

ID	016
Finding	TokenFilterMiddleware Documentation
Severity	0 - Informational
Description	The tokenFilterMiddleware type declares it inherits the `ICS4Wrapper` type, but it does not seem that is the case. Consider updating the documentation.
Recommendation	Update the documentation.
Code References	

QGB

Solidity

Finding 017 - An Erroneous _newValidatorSetHash Renders the Bridge Unusable

ID	017
Finding	An Erroneous `_newValidatorSetHash` Renders the Bridge Unusable
Severity	4 - LOW
Description	Validator set updates are made by calling `QuantumGravityBridge.updateValidatorSet` and passing in the `_newValidatorSetHash`. Validation of this hash occurs outside of the QuantumGravityBridge contract. In the event an erroneous hash is published, e.g. one missing address or an unsorted address array provided, future calls to both `updateValidatorSet` and `submitDataRootTupleRoot` will fail.
Recommendation	Compute the `_newValidatorSetHash` using `computeValidatorSetHash` as a precaution and include zero address checks when updating the validator set.
Code References	QuantumGravityBridge.sol#L247

Finding 018 - No Restrictions on `state_powerThreshold`

ID	018
Finding	No Restrictions on `state_powerThreshold`
Severity	4 - LOW
Description	`state_powerThreshold` may be set to any `uint256` value. If set too high, it becomes impossible for signature checks with `checkValidatorSignatures` to succeed.
Recommendation	When setting state_powerThreshold, validate it is being set within an acceptable range.
Code References	QuantumGravityBridge.sol#L52

Finding 019 - Unchecked Arithmetic Can Save Gas

ID	019
Finding	Unchecked Arithmetic Can Save Gas
Severity	0 - Informational
Description	Incrementing of `i` is certain to not overflow (block gas limit would be reached before an overflow is possible.
Recommendation	Make use of `unchecked {}` where operations are certain to not overflow.
Code References	QuantumGravityBridge.sol#L193-L226

Finding 020 - Differing Style in Use of state_eventNonce and state_lastValidatorSetCheckpoint

ID	020
Finding	Unchecked Arithmetic Can Save Gas
Severity	0 - Informational
Description	Both of `state_eventNonce` and `state_lastValidatorSetCheckpoint` are used one time, however, `state_eventNonce` is `sload'ed` prior to use whereas `state_lastValidatorSetCheckpoint` is `sload'ed` when needed.
Recommendation	Consider a consistent variable caching style performing both `sloads` at the beginning of the function.
Code References	QuantumGravityBridge.sol#L317

Go module

The Quantum Gravity Bridge module within the Celestia chain serves the purpose of providing attestations. These attestations can either signal Validator Set Changes or represent Data Commitments. The overall security is premised on the majority honesty of the Celestia validator power.

State Management vs. Validation

Context: Clarity and separation of concerns.
Issue: The module appears to mix state management functions with validation, blurring their distinct roles. Such a mix can lead to confusion and potential errors.
Example: The `GetLatestAttestationNonce` method panics instead of returning a boolean for nonce existence. This conflation means another function, `CheckLatestAttestationNonce`, is needed, complicating the logic.

Redundant Function Calls

Context: Inefficient usage of resources.
Issue: Some execution paths in the codebase call the same function multiple times, despite the output being already available.

Example: The method at abci.go gets the data commitment window, but then subsequently calls `NextDataCommitment` which fetches the data commitment window again.

Finding 021 - Key Prefix Type Issue

ID	021
Finding	Conformance with cosmos-sdk standards.
Severity	0 - Informational
Description	Key prefixes should utilize `bytes` instead of `string`
Recommendation	Convert key prefixes to `bytes` type.
Code References	keys.go

Finding 022 - Slow IsEVMAddress Function

ID	022
Finding	Over time, with a bloating state, the function's performance degrades.
Severity	5 - Medium
Description	The function's iteration over EVMAddresses in the QGB evm addr to val addr prefix results in reduced performance.
Recommendation	Integrate a KeySet (refer to cosmossdk.io/collections.KeySet) for faster existence verification.
Code References	keeper_valset.go

Finding 023 - Non-Idiomatic Code in genesis.go

ID	023
Finding	Improved code readability.
Severity	0 - Informational
Description	The `else if` structure affects code readability.
Recommendation	Replace `else if` with separate if conditions.
Code References	genesis.go

Finding 024 - Unused Variables in abi_consts.go

ID	024
Finding	Unused variables.
Severity	0 - Informational
Description	Variables present but not used.
Recommendation	Delete these variables or provide documentation explaining their presence.
Code References	• ExternalQGBABI • BridgeValidatorABI • DcDomainSeparator

Finding 025 - Unused Variables in errors.go

ID	025
Finding	Unused variables.
Severity	0 - Informational
Description	Presence of variables that aren't utilized.
Recommendation	Remove these variables or provide a rationale for their existence.
Code References	errors.go

Finding 026 - Unused ConvertByteArrToString in keys.go

ID	026
Finding	Unused function.
Severity	0 - Informational
Description	`ConvertByteArrToString` isn't used.
Recommendation	Delete this function or clarify its purpose.
Code References	keys.go

Finding 027 - Non-Idiomatic Bytes-to-String Conversion

ID	027
Finding	Code inefficiencies related to unnecessary conversions.
Severity	0 - Informational
Description	Non-idiomatic and inefficient string conversion. Resulting string data is converted back to bytes, causing unneeded copying.
Recommendation	Return bytes directly.
Code References	keys.go

Finding 028 - Sorting Inefficiency in validator.go

ID	028
Finding	Proper implementation of sorting for improved efficiency.
Severity	2 - Very Low
Description	The sort.Interface is not optimally implemented.
Recommendation	Use the `sort.Interface` correctly or utilize the `slices.Sort` package (if permitted by the Go version).
Code References	validator.go

Finding 029 - Inefficient Duplicate Handling

ID	029
Finding	The complexity of `HasDuplicates` is suboptimal.
Severity	2 - Very Low
Description	The function can be improved by using a map for existence checks, making it more idiomatic and efficient.
Recommendation	Refactor `HasDuplicates` to utilize a map for checking item existence.
Code References	validator.go

Finding 030 - Test Method in Production Code

ID	030
Finding	Proper separation of test and production code.
Severity	0 - Informational
Description	`GetPowers` method is used only in tests but exists in production code.
Recommendation	Move the method to test files.
Code References	validator.go

Finding 031 - Improper Use of Floats in PowerDiff

ID	031
Finding	Risks of non deterministic behaviour in float handling.
Severity	5 - Medium
Description	The method uses floats.
Recommendation	Switch to using `sdk.Dec`
Code References	validator.go

Finding 032 - Use of Floats in SignificantPowerDiff

ID	032
Finding	Risks of non deterministic behaviour in float handling.
Severity	5 - Medium
Description	Floats are used for the `SignificantPowerDiff` constant.
Recommendation	Transition to `sdk.Dec`.
Code References	abci.go

Finding 033 - Silent Overflow in PowerDiff

ID	033
Finding	Potential for incorrect results due to unnoticed overflows.
Severity	5 - Medium
Description	The function allows silent overflow.
Recommendation	Trigger a panic for overflows or use a type that supports conversion to `int64` without overflow.
Code References	validator.go

Finding 034 - Excessive Computation in PowerDiff

ID	034
Finding	Optimizing performance by reducing unnecessary calculations.
Severity	2 - Very Low
Description	The delta can be computed more efficiently during single validator power changes.
Recommendation	Optimize delta calculations.
Code References	validator.go

Finding 035 - TwoThirdThreshold Precision

ID	035
Finding	Improving precision in ratio calculations.
Severity	0 - Informational
Description	The returned ratio could be made more precise.
Recommendation	Enhance the precision of `TwoThirdThreshold` by applying a suitable multiplication factor.
Code References	valset.go

Finding 036 - Redundant Calculations in handleValSetRequest

ID	036
Finding	Optimizing computation by eliminating redundancy.
Severity	0 - Informational
Description	The function checks three conditions before the `SetAttestationRequest` branch, each with its computational overhead.
Recommendation	Sequentially check each condition and execute the branch once a condition evaluates to true.
Code References	abci.go

Finding 037 - Unoptimized Data Retrieval

ID	037
Finding	Performance optimization for data fetches.
Severity	0 - Informational
Description	The `GetDataCommitmentWindow` method unnecessarily goes through the gRPC server.
Recommendation	Use `keeper.GetParams`.
Code References	keeper_data_commitment.go

Finding 038 - Incomplete Genesis Export/Import

ID	038
Finding	Chain upgrades and state migrations.
Severity	0 - Informational
Description	Only Parameters are exported, not attestations.
Recommendation	Enable attestation export or provide documentation justifying its omission.
Code References	genesis.go

Finding 039 - Unexplained Module Consensus Version

ID	039
Finding	Chain upgrades and state migrations.
Severity	0 - Informational
Description	The consensus version of 2 is not documented, and no migration is registered.
Recommendation	Provide documentation or comments explaining this choice.
Code References	module.go