Binary Builders Preliminary Audit Report

Rollkit audit

Date of Engagement: Dec 11th, 2023 - Jan 10th, 2024

Document revision history

Version	Modification	Date	Author
0.1	Initial creation	2023-12-14	Onur Akpolat
0.2	Add Code Review	2024-01-05	Frojdi Dymylja
0.3	Add Code Review	2024-01-05	Aleksandr Bezobchuk
0.4	Review tweaks	2024-01-15	Frojdi Dymylja

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 December 11th, 2023 and ending on January 10th, 2024. The security assessment was scoped to the smart contracts and modules provided in the following Celestia GitHub repositories:

https://github.com/rollkit/rollkit/
Commit Hash: eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d

Audit Summary

The team was provided a little over two weeks 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

The audit of the Celestia Rollkit encompassed three modules: /block, /store, and /types. It identified several areas needing improvement across these modules, particularly in data storage strategies, naming conventions, code efficiency, and maintainability. The suggested enhancements focus on reducing complexity, improving documentation, ensuring efficient data handling, and enhancing code testability.

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	Description	Repository
block	This defines how block production and syncing for the rollup is handled.	https://github.com/rollkit/rollkit/tree/main/block
store	This defines how the rollup state is stored.	https://github.com/rollkit/rollkit/tree/main/store
types	This is where rollkit-specific types are defined.	https://github.com/rollkit/rollkit/tree/main/types

Audit Branch/Tag: **v0.11.0-rc2** (**05ff97ea78862631436c8b638d50a63de0e1b7b0)**
CometBFT (Tendermint) Version: **v0.38.0-rc3**
Tendermint Version): **v0.35.9**

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.

Assessment Summary & Findings Overview

The audit reveals a need for enhancing clarity, documentation, efficiency, and maintainability across the Rollkit Library. Addressing these issues will lead to a more robust, understandable, and efficient codebase, thereby improving the overall performance and reliability of the library.

- Refinement of Synchronization Techniques: The audit highlighted the need for improved synchronization mechanisms within the rollkit library. It was found that current usage of custom types like SyncerStatus and mechanisms for state synchronization in Manager and BlockCache were suboptimal. The recommended approach is to leverage standard library features, such as sync/atomic, for more efficient and maintainable code.

- Method Complexity and Lifecycle Management: Several methods, particularly in the Block module like BlockSyncService#Start and HeaderSyncService#Start, were identified as overly complex and lacking clear documentation. This complexity poses a risk for future debugging and maintenance. Simplification of these methods and better use of context handling to manage lifecycle events are recommended for clarity and better maintenance.

- Improving Data Handling and Code Practices: The audit pointed out several areas where data handling could be optimized, such as in GenerateKey and the handling of int64 overflows. Additionally, practices like storing testing methods within production code and inefficient mutex usage were noted. Enhancements in these areas are essential for better performance and reliability of the library.

Overall, these issues are predominantly informational and present a low risk to the rollkit library's functionality. However, addressing them will significantly enhance code quality, maintainability, and efficiency. Detailed recommendations have been provided for each finding to guide the improvement process.

Findings & Tech Details

Block

The /block module audit highlighted significant areas needing improvement, particularly in data storage strategies and method complexity. The module employs redundant synchronization techniques, as seen in the use of *SyncerStatus* and *lastState*.

Methods like *BlockSyncService#Start* and *HeaderSyncService#Start* are overly complex, increasing the risk for future maintenance challenges. Issues with context handling and concurrency, such as inefficient mutex usage, were also noted. The audit recommends adopting more standard synchronization methods, simplifying complex methods, and improving documentation and testing practices. These changes will enhance the module's maintainability and efficiency.

Finding 001 - Unnecessary Usage of SyncerStatus

ID	001
Finding	Unnecessary Usage of *SyncerStatus*
Severity	1 - Informational
Description	The *SyncerStatus* type is used to synchronize access to a boolean indicating the status of syncing. This proprietary type is not needed as you can just use the sync/atomic stdlib package to achieve the same thing, which results in less code being maintained.
Recommendation	Consider using sync/atomic stdlib in place of *SyncerStatus*.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/syncer_status.go

Finding 002 - PendingBlocks#getPendingBlocks Simplification

ID	002
Finding	*PendingBlocks#getPendingBlocks* Simplification
Severity	1 - Informational
Description	The method *getPendingBlocks* on the *PendingBlocks* creates a shallow copy of the blocks and sorts them. A manual copy of the slice can be avoided.
Recommendation	Consider calling the built-in *slices.Clone* method which returns a shallow copy instead.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/pending_blocks.go#L29-L32

Finding 003 - Context as a Field in HeaderSyncService and BlockSyncService

ID	003
Finding	Context as a Field in *HeaderSyncService* and *BlockSyncService*
Severity	2 - Informational
Description	Both the *HeaderSyncService* and *BlockSyncService* types have a Context field as part of the type. Many methods on these types accept a Context argument from the caller, which is ideal. Not only does this make the internal Context field unnecessary, it's also an indication of a codesmell and should be removed.
Recommendation	As most of the methods already accept Context as an argument, remove the Context field from both *HeaderSyncService* and *BlockSyncService*. For methods that do not accept a Context, update them to accept a Context from the caller in order to manage lifecycle events properly, e.g. *Start*.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/header_sync.go#L44 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/block_sync.go#L43

Finding 004 - BlockSyncService#Start Overly Complex

ID	004
Finding	*BlockSyncService#Start* Overly Complex
Severity	2 - Informational
Description	The *BlockSyncService#Start* method is rather complex and large. It is responsible for many tasks such as starting and subscribing to a subscriber, starting the block store, P2P logic, creating a syncer and evaluating P2P responses. In addition, the method is not very well documented. In the case this method is bug-prone or problematic or even needs to be updated in the future, it might be difficult for developers to truly understand and debug root causes.
Recommendation	We recommend splitting up the method into simpler, more testable methods, each that are clearly documented and well tested. In addition, *Start* should take a Context in order for the caller to manage the lifecycle of the call in a predictable manner.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/block_sync.go#L122

Finding 005 - HeaderSyncService#Start Overly Complex

ID	005
Finding	*HeaderSyncService#Start* Overly Complex
Severity	2 - Informational
Description	The *HeaderSyncService#Start* method is rather complex and large. It is responsible for many tasks such as starting and subscribing to a subscriber, starting the header store, P2P logic, creating a syncer and evaluating P2P responses. In addition, the method is not very well documented. In the case this method is bug-prone or problematic or even needs to be updated in the future, it might be difficult for developers to truly understand and debug root causes.
Recommendation	We recommend splitting up the method into simpler, more testable methods, each that are clearly documented and well tested. In addition, *Start* should take a Context in order for the caller to manage the lifecycle of the call in a predictable manner.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/header_sync.go#L121

Finding 006 - Use of lastState in Manager

ID	006
Finding	Use of *lastState* in *Manager*
Severity	3 - Informational
Description	The *Manager* type contains a reference to *lastState* object with an internal mutex to synchronize access to it. This lastState value is used in various critical places in the *Manager* API. In addition, the *Manager* also has a reference to *Store* which can get and set *State* as indicated by *Manager#updateState*. Having a direct reference to lastState in addition to store also storing this state can lead to footguns as you have two locations where this data is kept.
Recommendation	If there are situations where the state referenced by lastState and what Store contains, then that should be clearly documented. However, if there are no situations where they are expected to differ, then we recommend keeping access to the latest *State* via *Store*.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L58-L61

Finding 007 - Complexity of Manager and Insufficient Test Coverage

ID	007
Finding	Complexity of *Manager* and Insufficient Test Coverage
Severity	3 - Informational
Description	The *Manager* type contains mixed usage of multiple channels and a mutex to synchronize various aspects of the *Manager*. This leads to greater complexity in debugging, upgrading, and general understanding of the implementation. Especially around the notion of what data is protected and synchronized. Important methods such as *AggregationLoop*, *SyncLoop*, and *BlockStoreRetrieveLoop* are rather complex with little documentation on their function and how the channels are used within them. In addition, there seems to be a lack of test coverage for such an important component of the Rollkit codebase.
Recommendation	We see there is markdown documentation that explains the overall concepts and roles of each component and the general algorithm of the Manager and state management. However, we believe it would be prudent to improve the Godoc documentation of critical and public APIs of the *Manager* type. Specifically, outline the role and use of each channel and how Contexts come into play. In addition, consider refactoring some of the critical APIs to make them easier to understand and test. Finally, we recommend considerably increasing test coverage and edge case testing of the *Manager* type.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go

Finding 008 - Replace SyncStatus with atomic.Bool

ID	008
Finding	Replace SyncStatus with atomic.Bool
Severity	0 - Informational
Description	The RWMutex on sync status to protect a simple bool is overkill, and inefficient.
Recommendation	Replace with atomic.Bool
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/syncer_status.go#L9

Finding 009 - Contentious RWMutex usage in BlockCache

ID	009
Finding	Contentious RWMutex usage in BlockCache
Severity	0 - Informational
Description	Blocks, hashes and daIncluded maps share the same RWMutex, but they’re all accessed independently, each of the resources should have its own mutex.
Recommendation	Consider using a separate mutex for each map, or consider using some other primitive like sync.Map
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/block_cache.go#L14

Finding 010 - Inefficient usage of mutex in PendingBlocks

ID	010
Finding	Inefficient usage of mutex in PendingBlocks
Severity	0 - Informational
Description	The method `getPendinBlocks` sorts the slice whilst holding a mutex over it, which is unnecessary.
Recommendation	Move sorting out of the mutex lock.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/pending_blocks.go#L33

Finding 011 - Switch statement over IOTA without a default

ID	011
Finding	Switch statement over IOTA without a default
Severity	0 - Informational
Description	Switch statement over iota does not have a default case.
Recommendation	Add a default case in case, or simply use an IF
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L579

Finding 012 - Testing Methods in production code

ID	012
Finding	Testing Methods in production code
Severity	0 - Informational
Description	There are testing methods in production code, these testing methods also affect performance because a mutex lock is dependent on a component only used in testing.
Recommendation	Move the methods in a testing package, methods in testing packages can be accessed within testing context.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L208 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L213 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L220 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L225 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L230 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L235 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/block_cache.go#L13

Finding 013 - Unused doneBuildingBlock channel

ID	013
Finding	Unused doneBuildingBlock channel
Severity	0 - Informational
Description	The channel is created in NewManager, then it's reset every time in AggregationLoop, but never shared with other parts of the code.
Recommendation	Remove it
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L97 <- definition https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L192 <- local instantiation https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L295:L296 <- usage

Finding 015 - Timer is not Garbage Collected

ID	0XX
Finding	Timer is not Garbage Collected
Severity	0 - Informational
Description	Timer is not stopped when the Aggregation loop exits.
Recommendation	Add a defer timer.Stop statement.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L258

Finding 016 - Unchecked for closure txAvailable channel, might yield undesired publishing of blocks

ID	016
Finding	Unchecked for closure txAvailable channel, might yield undesired publishing of blocks
Severity	5
Description	The txAvailable channel is provided by the Mempool implementation, the mempool might decide to close the channel for whatever reason. This is not checked which means that the loop will become always (mainly) owned by the txAvailable channel closure yielding undesired and unneeded block publishing.
Recommendation	Check for channel closure
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L282

Finding 017 - Unchecked default DA response code might lead to extra loop cycles.

ID	017
Finding	Unchecked default DA response code might lead to extra loop cycles.
Severity	3
Description	If the DA Layer returns an unknown enum code, then the block publishing loop will do one extra cycle, if this is continuous the loop becomes endless.
Recommendation	Add a default erroring case on unknown enums.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L791

Finding 018 - Exhausting mutex access pattern in setDAIncluded

ID	018
Finding	Exhausting mutex access pattern in setDAIncluded
Severity	0 - Informational
Description	TODO
Recommendation	TODO
Code References	TODO: Link

Finding 019 - Unneeded work in publishBlock

ID	019
Finding	Unneeded work in publishBlock
Severity	0 - Informational
Description	If the node is not a proposer we immediately exit. We should do this immediately.
Recommendation	Do IsProposer check immediately.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L618

Finding 020 - Compute IsProposer once

ID	020
Finding	Compute IsProposer once
Severity	0 - Informational
Description	IsProposer can be computed at startup and that'd never change over the course of the program lifecycle.
Recommendation	Compute it immediately, since it's immutable.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L609

Finding 021 - Unsafe ctx.Done assumption in publishBlock

ID	021
Finding	Unsafe ctx.Done assumption in publishBlock
Severity	0 - Informational
Description	Between the select evaluation and the next channel send, the ctx might have become done.
Recommendation	Look finding 022
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L753 <- evaluation https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L760 <- when the CTX might have become done

Finding 022 - Context.Done is not honored during chan sends in publishBlock

ID	022
Finding	Context.Done is not honored during chan sends in publishBlock
Severity	0 - Informational
Description	Context.Done is not honored in chan sends.
Recommendation	Honor context.Done to avoid endlessly locking the operation. This might also create deadlocks. Select { Case <-ctx.Done: Case HeaderCh<-value }
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L760 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/block/manager.go#L763

Store

In the */store* module, the audit revealed opportunities for enhancing configuration accuracy and reducing redundancy. Notable issues include the misuse of Context within the *DefaultStore* type, which complicates lifecycle management, and the unnecessary usage of an external library for handling multi-errors.

Additionally, race conditions in methods like SetHeight and suboptimal usage of atomic operations were identified. Recommendations include refining the documentation for clarity, especially in functions like *NewDefaultInMemoryKVStore*, and improving error handling and thread safety. Implementing these changes will bolster the module's clarity and reliability.

Finding 023 - Document NewDefaultInMemoryKVStore options

ID	023
Finding	Document NewDefaultInMemoryKVStore options
Severity	1 - Informational
Description	The *NewDefaultInMemoryKVStore* function creates a data store with BadgerDB options that indicate usage for in-memory contexts. However, the hard-coded options use specific values without any context, which could lead to debugging oddities and inefficiencies down the line where it may not be clear why those specific values were chosen.
Recommendation	Document the rationale and/or purpose behind each option in the *badger3.Options* struct.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/kv.go#L19

Finding 024 - Context as a Field in DefaultStore

ID	024
Finding	Context as a Field in DefaultStore
Severity	2 - Informational
Description	The *DefaultStore* implementation of the *Store* interface contains a *Context* field. This Context is provided to further internal API calls which accept a Context as an argument. Typically, these Contexts are provided to manage the lifecycle of the called function. Providing it from a field of the *DefaultStore* type instead of having the caller provide it directly, seems like a codesmell and makes lifecycle management more difficult to control.
Recommendation	Consider removing Context as a field in the DefaultStore type and instead, allow the caller to establish a Context and provide it to the *Store* methods, i.e. modify the *Store* type to accept a *Context* as an argument in potentially time sensitive methods.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L33

Finding 025 - Unnecessary usage of external lib for multi-errors

ID	025
Finding	Unnecessary usage of external lib for multi-errors
Severity	1 - Informational
Description	The go.uber.org/multierr package is used to collect and chain errors in methods where multiple errors need to be handled as one. The Go stdlib allows for this functionality.
Recommendation	Consider using the Go *errors.Join* API over the *go.uber.org/multierr* package.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L81-L83

Finding 026 - Race Condition On SetHeight

ID	026
Finding	Race Condition on SetHeight
Severity	6 - Medium
Description	We get height using atomics, but do the comparison in a non synchronized step. As we do the comparison the value might have been changed by some other go-routine, making the CAS operation fail.
Recommendation	Fix the race condition maybe using a for loop: ``` for { storeHeight := atomic.LoadUint64(&s.height) if height <= storeHeight { break } if atomic.CompareAndSwapUint64(&s.height, storeHeight, height) { break } } ```
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L47

Finding 027 - Prefer atomic.Uint64 over manual atomic operations

ID	027
Finding	Prefer atomic.Uint64 over manual atomic operations
Severity	0 - Informational
Description	The Golang team advises to use atomic.Uint64 over manual atomic operations using functions.
Recommendation	Use atomic.Uint64
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L32 https://github.com/golang/go/blob/master/src/sync/atomic/doc.go#L106

Finding 028 - Refactor GenerateKey to be more robust

ID	028
Finding	Refactor GenerateKey to be more robust
Severity	0 - Informational
Description	GenerateKey simply uses `fmt.Sprintf("%v", field")` to convert a part of a multipart key into bytes. This is not robust, it is also most likely not computationally and space efficient. Also the API is unbounded by the usage of `interface{}` which means that also non-deterministic formattable values could be passed to it.
Recommendation	Make the API more robust, or constrain the accepted types to a smaller set.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/kv.go#L44

Finding 029 - Misuse of context

ID	029
Finding	Misuse of context
Severity	0 - Informational
Description	Context is meant to be scoped around the lifecycle of a single request, not the lifecycle of the entire program. Contexts are not designed for storing global or long-lived data. They are meant to be passed down the call chain and should not be stored in global or package-level variables.
Recommendation	Remove context from the DefaultStore struct, make the Store interface methods accept context.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L33 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/types.go#L21 <- Example of refactor where to introduce context as a param.

Finding 030 - Store Space Efficiency

ID	030
Finding	Store Space Efficiency
Severity	0 - Informational
Description	Store could favor storing information using uint64 (height) as key, and using the hash->height as a reverse index. This would yield lower key sizes, in fact: Storing uint64 as string: - best case: 1 byte - worst case: 20 byte (for 2^64-1) Storing hash (hex(sha256)): - constant 64bytes
Recommendation	Depends on access patterns. But consider refactoring storage of blocks and commits to use height as the primary key, and then retain an index (hash->height) to do the reverse mapping.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L212

Finding 031 - Unneeded work being done during SaveBlock

ID	031
Finding	Unneeded work being done during SaveBlock
Severity	0 - Informational
Description	If one of the put operations fails, then no more attempts at `Put`ting should be done.
Recommendation	Exit when one operation fails, since in the end the tx is discarded if one Put op fails.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/store/store.go#L82

Types

The audit of the */types* module in the Celestia Rollkit Library identified several areas for improvement, focusing on validation and data handling. Key issues include incomplete verification in the *Header#Verify* method and potential integer overflows when converting between *uint64* and *int64*. The library also exhibits inefficiencies like computing the same empty evidence hash repeatedly.

It is recommended to enhance validation procedures, particularly in the *Commit#Validate* method, and to optimize data handling practices to prevent overflows and redundant computations. Addressing these issues will improve the module's robustness and efficiency.

Finding 032 - Header Verification

ID	032
Finding	Header Verification
Severity	2 - Informational
Description	The *Verify* method defined on the *Header* type is lacking any verification or validation outside of checking the aggregator hashes. It's not clear if this method is expected to verify more, but there is an unresolved TODO without any reference issue link. This might add to technical debt as this particular part of the codebase evolves.
Recommendation	Consider removing the TODO in what seems to be a critical method on the *Header* type and/or consider expanding upon verification and validation of the header compared to the untrusted header, such as *ChainID* and *Version*. At the very least, there should be a clear godoc on the method that describes what verification is expected to perform.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/header.go#L82

Finding 033 - Possible Int64 overflows

ID	033
Finding	Various possible overflows, caused by unchecked convertions
Severity	0 - Medium
Description	When converting from uint64 to int64 header height.
Recommendation	Since the function is allowed to return an error, do check if it is in int64 bounds.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/abci/block.go#L20 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/abci/block.go#L48 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/block.go#L64 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/hashing.go#L17 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/header.go#L79

Finding 034 - Unnecessary computing of empty evidence hash.

ID	034
Finding	EvidenceHash could be constant
Severity	0 - Informational
Description	We compute the evidence hash of the same evidence variable which is always empty, this could be computed constantly.
Recommendation	Consider creating a global, unexported variable of the empty evidence hash.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/abci/block.go#L34 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/abci/block.go#L62 https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/hashing.go#L31

Finding 035 - Incomplete Validation of Commit

ID	035
Finding	Incomplete validation of commit
Severity	0 - Medium
Description	Commit Validation only asserts if one signature is present, but does not assert if that signature is valid, or even different from nil.
Recommendation	Add some minimal validation on the single signature.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/block.go#L105

Finding 036 - Incorrect Untrusted Header height validation

ID	036
Finding	Incorrect Untrusted Header height validation
Severity	0 - Informational
Description	Untrusted header height is expected to be >= the trusted header but in reality it should just be == to trusted header +1.
Recommendation	Fix the validation step to do an equality check.
Code References	https://github.com/rollkit/rollkit/blob/eccdd0f1793a5ac532011ef4d896de9e0d8bcb9d/types/signed_header.go#L54