

INTERNAL QA REPORT

Rollups Node v2.0.0-alpha.12: internal security review

Node, contracts and CLI, tested on a devnet and on Base Sepolia and OP Sepolia

Target	cartesi/rollups-node v2.0.0-alpha.12, cartesi/rollups-contracts v3.0.0-alpha.6, @cartesi/cli alpha.34 and alpha.35
Cycle	alpha.12
For	rollups-node, rollups-contracts and CLI maintainers, and the Cartesi developer community
Prepared by	Cartesi QA team
Date	2026-08-04
Status	Final

Published by Cartesi in the documentation. This is an internal review by the Cartesi QA team. It is not an audit by an external firm.

Contents

Where to start	3
Scope and method	3
Findings	4
ALPHA12-01 Host wall-clock deadline of an advance is committed as a processed input	5
ALPHA12-02 Deposit credits the requested amount, not the amount received	5
ALPHA12-04 Multiple inputs in one transaction stall the application	6
ALPHA12-03 Foreclosure before the first accepted claim traps early deposits	7
ALPHA12-06 Withdrawal config accepts a zero guardian or a zero output builder	7
ALPHA12-05 CLI hardcodes a 30,000,000 gas limit	8
ALPHA12-07 Deposits of ERC-20 tokens that return no value cannot be made	8
Developer experience observations	9
Evidence and links	9
Appendix: how to read the ratings	10

Where to start

Ordered by severity. Click an ID to jump to the finding, or open its links (PR, write-up) directly.

ID	Finding	Severity	Owner	Write-up
ALPHA12-01	Host wall-clock deadline of an advance is committed as a processed input	Critical	rollups-node team	Deadline check in the machine Status mapping in the manager
ALPHA12-02	Deposit credits the requested amount, not the amount received	High	rollups-contracts team	ERC20Portal at the reviewed version
ALPHA12-04	Multiple inputs in one transaction stall the application	High	rollups-node team	Initial schema at the reviewed version
ALPHA12-03	Foreclosure before the first accepted claim traps early deposits	Medium	rollups-contracts team	Application at the reviewed version
ALPHA12-06	Withdrawal config accepts a zero guardian or a zero output builder	Medium	rollups-contracts team	LibWithdrawal Config at the reviewed version
ALPHA12-05	CLI hardcodes a 30,000,000 gas limit	Low	rollups-node team	Issue rollups-node#786 Gas limit constant at the reviewed version
ALPHA12-07	Deposits of ERC-20 tokens that return no value cannot be made	Low	rollups-contracts team	ERC20Portal at the reviewed version SafeERC20 helper at the reviewed version

Scope and method

What was tested

Component	Version / commit	Environment	Notes
rollups-node	v2.0.0-alpha.12	local devnet, Base Sepolia, OP Sepolia	advancer, claimer, evm-reader, validator, JSON-RPC API, CLI
rollups-contracts	v3.0.0-alpha.6	local devnet, Foundry tests	portals, application, consensus, withdrawal configuration
machine guest tools and libcmnt	v0.17.2	local devnet	guest side of the input and output paths
Cartesi Machine emulator	v0.20.0	local devnet	as embedded in the node
@cartesi/cli	alpha.34 and alpha.35	local devnet	create, build and run

Not covered

- Dave and PRT (fraud proofs), which were out of scope.
- The five exploratory charters of the validation catalog (CHR-001 to CHR-005: claimer misconfiguration, evm-reader stress, CLI misuse, day-two operations, multi-app) were not run.

Method

Source review of the node services (advancer, claimer, evm-reader, validator and JSON-RPC API), the CLI and the contracts.

A validation test catalog executed against a local devnet and public testnets, covering the CLI, inputs, egress, configuration, state persistence and services.

Targeted proofs of concept for each suspected issue, run against the real components.

Fidelity to mainnet

The review assumed a single-operator Authority consensus, a third-party HTTPS RPC provider, and an application that moves assets (Ether and tokens) through the portals. Reproductions ran on a local devnet and on public testnets, and ratings describe the outcome on a production deployment that follows the documented defaults.

Findings

Grouped by severity. Each block is a summary; the full write-up has the proof, the control experiment and the source references.

Critical (1)

ALPHA12-01 | Host wall-clock deadline of an advance is committed as a processed input

Severity	Owner	Component
Critical	rollups-node team	rollups-node, advancer and machine manager
Network		Confidence
chain-agnostic		high

Impact. An advance is aborted on a host wall-clock deadline (`AdvanceMaxDeadline`, 180 seconds by default) instead of a deterministic guest cycle budget. When it fires, the node records the input as processed with a time-limit status and no error, keeps the previous machine and output roots, discards the outputs of that input, and may finalize the epoch. Byte-identical inputs can therefore produce different final claims depending only on host speed, host load or the configured deadline. With a single-operator Authority the signed claim is final, so other nodes cannot reproduce it, and a deposit credit or withdrawal voucher emitted just before the deadline is silently discarded while the input still counts as processed, so funds can be lost or locked.

Evidence.

- `pkg/machine/implementation.go:434` compares `time.Since(startTime)` with the run timeout, set from `AdvanceMaxDeadline` at line 387.
- `internal/manager/instance.go:352-367` keeps the previous hash for a non-accepted input but still increments the processed-input counter; `:753-754` maps the deadline error to a time-limit status with a nil error.
- Observed on a devnet with two nodes sharing one application and one input: different node-local deadlines produced different machine hashes and output roots, and one node finalized a claim the other could not reproduce.

Suggested fix. Do not derive a committed completion status from a host wall-clock deadline. Treat a watchdog expiry as an infrastructure failure, and let only a deterministic guest cycle bound change the committed roots.

Full write-up: [Deadline check in the machine](#) | [Status mapping in the manager](#)

High (2)

ALPHA12-02 | Deposit credits the requested amount, not the amount received

Severity	Owner	Component
High	rollups-contracts team	rollups-contracts, ERC20Portal
Network		Confidence
conditional		high

Impact. `depositERC20Tokens` encodes the caller-supplied value into the input after `transferFrom`, without measuring how many tokens the application actually received. Tokens that charge a fee on transfer, rebase or run hooks deliver less than the value, while the input tells the application to credit the full value. The application over-credits the depositor, the pooled on-chain balance becomes smaller than the sum of the credits, and later withdrawals can fail. A depositor can use this to inflate an in-app balance relative to the real assets. It needs a token with this behavior, so the network is rated conditional.

Evidence.

- `src/portals/ERC20Portal.sol:22-37` transfers and then encodes value;
`src/common/InputEncoding.sol:38-50` builds the payload. There is no balance delta check in the deposit path.
- Observed with a Foundry test and a mock token charging a 1% fee: a deposit of 1000 delivered 990 tokens to the application while the input credited 1000.

Suggested fix. Measure the application balance before and after the transfer and reject the deposit unless the delta equals the value.

Full write-up: [ERC20Portal at the reviewed version](#)

ALPHA12-04 | Multiple inputs in one transaction stall the application

Severity	Owner	Component
High	rollups-node team	rollups-node, evm-reader and repository
Network		Confidence
chain-agnostic		high

Impact. Two or more `InputBox.addInput` calls in the same transaction were stored with the same `transaction_reference` (the transaction hash). This violated the `input_application_id_tx_reference_unique` constraint, and the evm-reader rolled back and retried the same block indefinitely. After such a transaction was included on chain, the affected application stopped indexing new inputs and could not recover without manual intervention. Any contract or user able to add several inputs in one transaction, for example a batcher or an aggregator, could trigger it, so the condition was reachable without special permissions.

Evidence.

- Found by validation test INP-006 on a testnet, then reproduced from scratch on a devnet with a template application.
- The uniqueness constraint is defined in the initial schema migration of the repository package.

Suggested fix. Identify inputs by transaction hash and log index instead of the transaction hash alone.

Full write-up: [Initial schema at the reviewed version](#)

Medium (2)

ALPHA12-03 | Foreclosure before the first accepted claim traps early deposits

Severity	Owner	Component
Medium	rollups-contracts team	rollups-contracts, Application and consensus
Network		Confidence
chain-agnostic		high

Impact. The proof of the accounts drive requires the computed machine root to equal the last finalized machine root, which is zero until the first claim is accepted. If an application is foreclosed before that first acceptance, there is no finalized root to prove against. The accounts drive proof can never succeed, account validation always reverts, and deposits made before the first finalization cannot be withdrawn through the emergency withdrawal. It is rated medium because it needs a foreclosure in the window before the first claim.

Evidence.

- `src/dapp/Application.sol:172-210` and `src/consensus/AbstractConsensus.sol:79-86, 186-189`.
- Observed with a Foundry test: a deposit followed by a foreclosure before any accepted claim made the proof revert.

Suggested fix. Give the pre-first-claim window a defined recovery path, for example a refund of deposits that were never finalized.

Full write-up: [Application at the reviewed version](#)

ALPHA12-06 | Withdrawal config accepts a zero guardian or a zero output builder

Severity	Owner	Component
Medium	rollups-contracts team	rollups-contracts, LibWithdrawalConfig and Application
Network		Confidence
chain-agnostic		high

Impact. The `isValid()` check only validates the geometry of the accounts drive. It does not reject a guardian or a withdrawal output builder equal to the zero address, and the factories forward the configuration unchanged. With a zero guardian, `foreclose()` can never be called, so the emergency withdrawal cannot be triggered. With a zero output builder, `withdraw()` reverts for every account. Funds deposited in such an application cannot be recovered through the emergency withdrawal if the consensus stops producing claims. It is rated medium because the guardian and the builder are chosen by the application developer at deployment, a developer can deliberately choose an application without emergency withdrawal, and applications deployed with a valid guardian and builder are not affected.

Evidence.

- `src/library/LibWithdrawalConfig.sol:10-35`, used by the constructor at `src/dapp/Application.sol:131-143`.
- Observed with a Foundry test: an application deployed through the factory with a zero guardian, and separately with a zero builder, was accepted, and `foreclose()` and `withdraw()` reverted.

Suggested fix. Reject a zero guardian and a zero output builder in `isValid()`, unless a documented design allows an application without emergency exit.

Full write-up: [LibWithdrawalConfig at the reviewed version](#)

Low (2)

ALPHA12-05 | CLI hardcodes a 30,000,000 gas limit

Severity	Owner	Component
Low	rollups-node team	rollups-node, cartesi-rollups-cli and the ethutil package
Network		Confidence
conditional		high

Impact. Every transaction sent by the CLI (`send` and `execute`) was created with a fixed gas limit of 30,000,000, which disables gas estimation. Networks that cap gas per transaction below this value, such as Base Sepolia and OP Sepolia, rejected the transaction before it reached a block with `exceeds max transaction gas limit`. The operator CLI could not send inputs or execute vouchers on those networks. This affects availability of the tooling, not the safety of funds.

Evidence.

- `pkg/ethutil/ethutil.go:23-24` defines the constant and `pkg/ethutil/transaction.go:46-68` applies it to every transaction.
- Found by validation tests ILC-005, ILC-006 and ILC-008 on a testnet.

Suggested fix. Leave the gas limit at zero so the transaction is estimated, and offer an optional override flag.

Full write-up: [Issue rollups-node#786](#) | [Gas limit constant at the reviewed version](#)

ALPHA12-07 | Deposits of ERC-20 tokens that return no value cannot be made

Severity	Owner	Component
Low	rollups-contracts team	rollups-contracts, ERC20Portal
Network		Confidence
chain-agnostic		high

Impact. The portal calls `transferFrom` and expects a boolean in the return data. Tokens that return nothing from `transferFrom`, such as USDT, make that call `revert`, while the withdrawal side of the contracts uses the OpenZeppelin `SafeERC20` helper. Deposits of those tokens through the portal always `revert`. The `revert` is atomic, so no funds are lost, but applications cannot accept a class of widely used tokens through the canonical portal. It is rated low because the portal documentation states that only ERC-20 compliant tokens are supported and that empty or ill-formed return values are not accepted.

Evidence. `src/portals/ERC20Portal.sol:28` uses the raw call, while `src/delegatecall/SafeERC20Transfer.sol:6-17` uses `SafeERC20`.

Suggested fix. Use `SafeERC20.safeTransferFrom` in the portal, or keep the behavior and document the supported token set prominently.

Developer experience observations

The validation run also produced usability observations, which were passed to the maintainers. They are not counted among the 7 findings above.

Test	Component	Observation
CLI-002	CLI	<code>cartesi</code> build with a missing dependency prints a JavaScript stack trace together with the error.
JRP-001	JSON-RPC API	Malformed JSON is answered with HTTP 400 and a plain-text body instead of a JSON-RPC error object with code -32700. The parse error constant is defined and never emitted.
JRP-003	JSON-RPC API	A wrong parameter type returns code -32602 with a generic message and no data. The detail that names the offending parameter is only logged at debug level.
CFG-005	Configuration	An invalid database connection string fails after about 15 seconds with a generic context deadline error that does not name the host.

Evidence and links

Every link in this report, in one place. Private repositories need access.

Ref	What	URL
ALPHA12-01	Deadline check in the machine	https://github.com/cartesi/rollups-node/blob/v2.0.0-alpha.12/pkg/machine/implementation.go
ALPHA12-01	Status mapping in the manager	https://github.com/cartesi/rollups-node/blob/v2.0.0-alpha.12/internal/manager/instance.go
ALPHA12-02	ERC20Portal at the reviewed version	https://github.com/cartesi/rollups-contracts/blob/v3.0.0-alpha.6/src/portals/ERC20Portal.sol
ALPHA12-04	Initial schema at the reviewed version	https://github.com/cartesi/rollups-node/blob/v2.0.0-alpha.12/internal/repository/postgres/schema/migrations/000001_create_initial_schema.up.sql
ALPHA12-03	Application at the reviewed version	https://github.com/cartesi/rollups-contracts/blob/v3.0.0-alpha.6/src/dapp/Application.sol
ALPHA12-06	LibWithdrawalConfig at the reviewed version	https://github.com/cartesi/rollups-contracts/blob/v3.0.0-alpha.6/src/library/LibWithdrawalConfig.sol
ALPHA12-05	Issue rollups-node#786	https://github.com/cartesi/rollups-node/issues/786
ALPHA12-05	Gas limit constant at the reviewed version	https://github.com/cartesi/rollups-node/blob/v2.0.0-alpha.12/pkg/ethutil/ethutil.go

Ref	What	URL
ALPHA12-07	SafeERC20 helper at the reviewed version	https://github.com/cartesi/rollups-contracts/blob/v3.0.0-alpha.6/src/delegatecall/SafeERC20Transfer.sol

Appendix: how to read the ratings

Severity describes the realistic outcome on a mainnet deployment, not on the environment where the finding was reproduced. Where reasoning and reproduction differ in parameters, the finding says what changes with the mainnet value. When in doubt between two levels the lower one is used, and a reviewer can argue it up.

Severity	Meaning
Critical	Loss of user or application funds, reachable by an outside party, with no operator action able to prevent it once triggered.
High	A wrong on-chain result is finalized without funds moving, or the application stops permanently (no automatic recovery; needs a redeploy or manual intervention), reachable by an outside party or by a common operational fault.
Medium	Funds or liveness are affected, but an operator can recover without redeploying.
Low	Operational nuisance: wrong health status, noisy logs, resource growth that only matters at scale, an inconvenient but safe default.
Info	Not a bug: a design tradeoff worth documenting, a near-miss that did not reproduce, or a coverage gap with no known failure.

Field	Values
Network	chain-agnostic (does not depend on the network), conditional (depends on a nameable network property or non-default config), devnet-only (reproduces on a test devnet but would not change the outcome with realistic parameters; capped at Low).
Confidence	in the conclusion: high (root cause traced in source and the mainnet outcome reproduced or clearly settled), medium (one of those is missing), low (early signal).