PROTOFIRE

February 2026

Smart contract audit
rain.intorastring

Content

Project description

The rain.intorastring repository implements a specialized Solidity library, LibIntOrAString, designed to represent strings as a single packed 32-byte EVM word (uint256). This IntOrAString type is optimized for extreme gas efficiency and constrained storage environments (like Rainlang stack values). It allows strings up to 31 bytes in length to be stored directly on the stack or in a single storage slot. The library handles the packing and It allows strings up to 31 bytes in length to be stored directly on the stack or in a single storage slot. The library handles the packing and handling.

Executive summary

Type	Library
Languages	Solidity
Methods	Architecture Review, Manual Review, Unit Testing, Functional Testing, Automated Review
Documentation	README.md
Repositories	https://github.com/rainlanguage/rain.intorastring

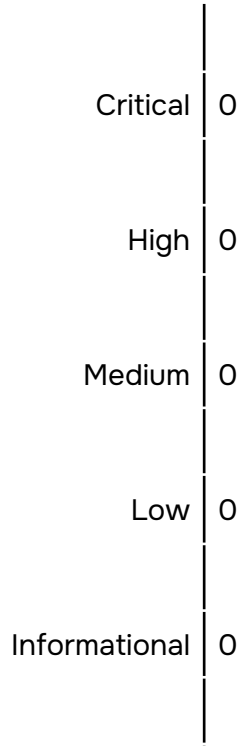
Reviews

Date	Repository	Commit
05/02/26	rain.intorastring	8e5f0db1b84fc746b736368bd12f08fc5392bcfe

Scope

Contracts
src/lib/LibIntOrAString.sol

Technical analysis and findings



Security findings

**** Critical

No critical severity issue found.

*** High

No high severity issue found.

** Medium

No medium severity issue found.

* Low

No low severity issue found.

Informational

No informational severity issue found.

General Risks

- Silent Truncation: The library strictly enforces a 31-byte capacity by truncating input strings modulo 32 (e.g., a 32-byte string becomes a 0-byte string) in the *fromStringV3()* function. This behavior is intentional and silent; there are no reverts or error messages. Consumers are solely responsible for validating input lengths to prevent unintended data loss.

Approach and methodology

To establish a uniform evaluation, we define the following terminology in accordance with the OWASP Risk Rating
Methodology:

	**Likelihood** Indicates the probability of a specific vulnerability being discovered and exploited in real-world scenarios
	**Impact** Measures the technical loss and business repercussions resulting from a successful attack
	**Severity** Reflects the comprehensive magnitude of the risk, combining both the probability of occurrence (likelihood) and the extent of potential consequences (impact)

Likelihood and impact are divided into three levels: High H, Medium M, and Low L. The severity of a risk is a blend of these two factors, leading to its classification into one of four tiers: Critical, High, Medium, or Low.

When we identify an issue, our approach may include deploying contracts on our private testnet for validation through testing. Where necessary, we might also create a Proof of Concept PoC to demonstrate potential exploitability. In particular, we perform the audit according to the following procedure:

	**Advanced DeFi Scrutiny** We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs
	**Semantic Consistency Checks** We then manually check the logic of implemented smart contracts and compare with the description in the white paper.
	**Security Analysis** The process begins with a comprehensive examination of the system to gain a deep understanding of its internal mechanisms, identifying any irregularities and potential weak spots.