PROTOFIRE

February 2026

Smart contract audit
rain.tofu.erc20-decimals

Content

Project description

This repository code is designed to read and store ERC20 token decimals using a "Trust On First Use" (TOFU) approach. Since the decimals() function is optional in the ERC20 standard and technically mutable, this system mitigates risk by reading the value once and storing it. Subsequent reads verify that the current token decimals match the stored value, protecting downstream logic (like fixed-point conversions) from unexpected changes.

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.tofu.erc20-decimals

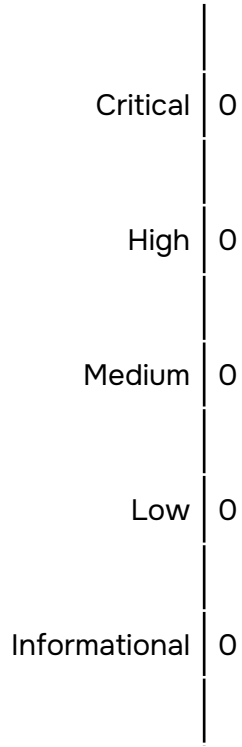
Reviews

Date	Repository	Commit
10/02/26	rain.tofu.erc20-decimals	5789bad61299b818000eb80d20f61a09a076d78c
11/02/26	rain.tofu.erc20-decimals	03406b6ad8d33a802c6eb58ae4dac3f0ea9aed38

Scope

Contracts
src/concrete/TOFUTokenDecimals.sol
src/interface/ITOFUTokenDecimals.sol
src/lib/LibTOFUTokenDecimals.sol
src/lib/LibTOFUTokenDecimalsImplementation.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 and Assumptions

- Trust on First Use (TOFU) Assumption: The system fundamentally relies on the assumption that a token's decimals should not change. If a legitimate token upgrades or changes its decimals after the first read, the stored value will become "Inconsistent," causing safe reads to revert and potentially locking functionalities that depend on it.

- ERC20 Non-Standard Behavior: While decimals() is widely adopted, it is optional in the EIP-20 specification. Tokens that do not implement it, or implement it in a non-standard way (e.g., returning values > 255), will result in a ReadFailure.

- Denial of Service (DoS) via Inconsistency: If a token is malicious or broken and starts returning a different decimal value than what was initially stored, calls to safeDecimalsForToken will revert. This provides a safety guarantee but could lead to a Denial of Service for any protocol interactions involving that specific token.

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.