SMART CONTRACT AUDIT REPORT

Radicle Drips

Author - Satyam Agrawal

License

Disclaimer

THE CONTENT OF THIS AUDIT REPORT IS PROVIDED “AS IS”, WITHOUT REPRESENTATIONS AND WARRANTIES OF ANY KIND.

THE AUTHOR DISCLAIM ANY LIABILITY FOR DAMAGE ARISING OUT OF, OR IN CONNECTION WITH, THIS AUDIT REPORT.

Introduction

Purpose of This Report

Author has been engaged by Radicle Foundation to perform a security audit of the Radicle drips hub contract codebase

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebases Submitted for the Audit

The audit has been performed on the following GitHub repositories:

Repository	Commit hash
https://github.com/radicle-dev/drips-contracts	`835656a99015e3cc28ee1003924654e5071f3d00`

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

The Radicle Drips Hub contract enables the user to define drips or a stream of money to its recipients on a specified cadence. By dividing the entitled funds among themselves, recipients can also give their entitled drips to others.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits, and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. To help evaluate the remaining risk, we measure the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

High complexity or low test coverage does not necessarily equate to a higher risk. However, certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Summary of Findings

No	Description	Severity	Status
1	Set Splits would be lost the splittable value for old split receivers.	Major	Acknowledged
2	Inefficient mathematical operations	Informational	Acknowledged
3	Inconsistent naming convention used in the codebase.	Informational	Acknowledged

Detailed Findings

1. Set Splits would be lost the splittable value for old split receivers.

Severity: Major

Context: DripsHub.sol#L459

`setSplits` function can be called independently of the split function, but if there is already some value that has been ready to split but `split` function would not be explicitly called on-chain. If the user again calls `setSplits` with new receiver sets, then old receivers will not be able to receive split funds anymore.

Recommendation

We recommend calling the `split` function before calling `setSplits`.

Status: Acknowledged

Client Comment: This is the designed and expected characteristic of the protocol.

2. Inefficient mathematical operations

Severity: Informational

Context: Drips.sol#274

The drips-receiving procedure will be facilitated by frequent calls to `_receiveDripsResult()`. For a frequent drips receiver user, even tiny gas savings can add up to significant monetary savings. Due to the fact that `receivableCycles` and `toCycle` cannot go underflow, it is unnecessary to waste gas doing inherent underflow and overflow checks.

There are other multiple places that can be found in the codebase to save gas like this.

Recommendation

We recommend using `unchecked` blocks to avoid inherent underflow and overflow checks on mathematical operations.

Status: Acknowledged

Client Comment: Worth looking into across the entire protocol.

3. Inconsistent naming convention used in the codebase.

Severity: Informational

Context: DripsHub.sol#239

The term `userId` is used throughout the codebase to identify the setter or receiver of drips as well when the person is doing an operation on its receivable drips. While the `receiver` variable name is used to specify the recipient of drips. But in this situation, `userId` is used for the recipient and `senderId` for the person who initiated the drips. As a result code readability is reduced and creates confusion.

Recommendation

It is preferable to have uniformity across the codebase. Use `userId` for the person who set the drips and `receiver` or `receiverId` for the person for whom the drips are set.

Status: Acknowledged

Client Comment: Need some time to decide whether we want to alter the convention or not.