Security Audit Report

aPriori APR Token and Staked APR Token

v1.0

October 3, 2025

Table of Contents

License

Disclaimer

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

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

THIS AUDIT REPORT WAS PREPARED EXCLUSIVELY FOR AND IN THE INTEREST OF THE CLIENT AND SHALL NOT CONSTRUE ANY LEGAL RELATIONSHIP TOWARDS THIRD PARTIES. IN PARTICULAR, THE AUTHOR AND HIS EMPLOYER UNDERTAKE NO LIABILITY OR RESPONSIBILITY TOWARDS THIRD PARTIES AND PROVIDE NO WARRANTIES REGARDING THE FACTUAL ACCURACY OR COMPLETENESS OF THE AUDIT REPORT.

FOR THE AVOIDANCE OF DOUBT, NOTHING CONTAINED IN THIS AUDIT REPORT SHALL BE CONSTRUED TO IMPOSE ADDITIONAL OBLIGATIONS ON COMPANY, INCLUDING WITHOUT LIMITATION WARRANTIES OR LIABILITIES.

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security GmbH has been engaged by aPriori Network INC. to perform a security audit of aPriori APR Token and Staked APR Token.

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).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/apriori-network/apr-token
Label	`apr-token`
Commit	`e3716910f95e5685ee79da79b32b411c44b4c792`
Scope	All contracts were in scope.

Repository	https://github.com/apriori-network/apr-staking-contracts
Label	`apr-staking-contracts`

Commit	`7d9a8244589223c22a38075ff53ccb0f74fee6bd`
Scope	All contracts were in scope.
Fixes verified at commit	`e2bb72288263bbca8c9905286a56edeb31fe7c7a` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

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 APRToken is an upgradeable ERC20 contract that, on initialization, assigns an admin address and mints a fixed initial supply to a specified recipient; beyond that, it behaves like a standard fungible token.

The StakedAPR is an upgradeable tokenized vault that takes deposits of an underlying asset, issues shares, and uses a cooldown-based redemption process where users request withdrawals and later claim from a dedicated pool; rewards can be added by an authorized role and vest linearly over time, with unvested amounts excluded from accounting, and key parameters (cooldown and vesting) configurable by the admin.

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**, **Partially Resolved**, 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. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

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

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	**Low**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**High**	Each function of the contracts was documented.
Test coverage	**Medium-High**	The test coverage reported by `forge coverage` is: ● `apr-token: 100%` ● `apr-staking-contracts: 98,10%`

Summary of Findings

No	Description	Severity	Status
1	The unstake function allows zero-share calls and resets cooldown in a way that may delay withdrawals	Minor	Resolved
2	Centralization risks from privileged roles	Minor	Acknowledged
3	Documentation inconsistency in `IStakedAPR` Interface	Informational	Resolved

Detailed Findings

1. The `unstake` function allows zero-share calls and resets cooldown in a way that may delay withdrawals

Severity: Minor

In `apr-staking-contracts:src/StakedAPR.sol:205-219`, the `unstake` function allows users to unstake their tokens. However, it exhibits two problematic behaviors.

First, it permits calls with `shares` equal to zero, which results in no assets being unstaked but still updates `claimableAt` to the current timestamp plus the cooldown duration.

This means that previously unstaked funds that were already progressing toward maturity are forced back into a fresh cooldown, creating a self-inflicted penalty and a poor user experience.

Second, the function aggregates all unstake requests into a single cooldown record. Each new call overwrites `claimableAt`, which causes the cooldown for the entire position to restart.

As a result, if a user adds a small unstake request after a larger one, all funds are delayed even if part of them was close to becoming claimable.

Recommendation

We recommend rejecting zero-effect calls by requiring a strictly positive `shares` value. In addition, a more user-friendly cooldown design could be implemented, either by tracking each unstake as a separate tranche with its own maturity

Status: Resolved

2. Centralization risks from privileged roles

Severity: Minor

The protocol places significant authority in a small set of privileged roles, which introduces centralization concerns.

The StakedAPR administrator can immediately adjust critical parameters such as the cooldown duration and vesting period, without the protection of a timelock.

The `REWARDER_ROLE` holds complete control over when and how rewards are distributed, leaving no checks on timing or amounts.

In addition, the StakedAPR contract itself fully controls all assets during the cooldown period, with no distribution of responsibility.

There are no multi-signature requirements, meaning a single account can execute all administrative operations. This amplifies the risk of misuse or compromise of an administrator key.

Furthermore, administrators can unilaterally toggle between cooldown and no-cooldown operation, changing withdrawal behavior for all users. While the APRToken administrator cannot mint new tokens, they retain control over role assignments, consolidating governance power.

Overall, the current structure centralizes authority, relying heavily on trust in a few privileged accounts.

Recommendation

We recommend implementing timelock contracts for parameter changes, multi-signature wallets for admin roles, and clear documentation of all privileged functions.

Status: Acknowledged

3. Documentation inconsistency in `IStakedAPR` Interface

Severity: Informational

In `apr-staking-contracts:src/interfaces/IStakedAPR.sol:26-32`, the function documentation for `setCooldownDuration` references `cooldownShares` and `cooldownAssets` methods.

However, these methods are not implemented within the interface or the codebase.

This discrepancy between documentation and actual functionality may mislead developers or integrators, creating false assumptions about available features.

Recommendation

We recommend updating the documentation to accurately reflect the implemented functionality.

Status: Resolved