Safe Token

Airdrop

by Ackee Blockchain

August 26, 2022

Contents

1. Document Revisions

0.1	Draft report	July 18, 2022
1.0	Final report	July 18, 2022
1.1	Fix review	August 25, 2022
1.2	Change the title	August 26, 2022

2. Overview

This document presents our findings in reviewed contracts.

2.1. Ackee Blockchain

Ackee Blockchain is an auditing company based in Prague, Czech Republic, specialized in audits and security assessments. Our mission is to build a stronger blockchain community by sharing knowledge – we run a free certification course Summer School of Solidity and teach at the Czech Technical University in Prague. Ackee Blockchain is backed by the largest VC fund focused on blockchain and DeFi in Europe, Rockaway Blockchain Fund.

2.2. Audit Methodology

1. **Technical specification/documentation** - a brief overview of the system is requested from the client and the scope of the audit is defined.
2. **Tool-based analysis** - deep check with automated Solidity analysis tools and Slither is performed.
3. **Manual code review** - the code is checked line by line for common vulnerabilities, code duplication, best practices and the code architecture is reviewed.
4. **Local deployment + hacking** - the contracts are deployed locally and we try to attack the system and break it.
5. **Unit and fuzzy testing** - run unit tests to ensure that the system works as expected, potentially write missing unit or fuzzy tests.

2.3. Review team

Member's Name	Position
Jan Kalivoda	Lead Auditor
Josef Gattermayer, Ph.D.	Audit Supervisor

2.4. Disclaimer

We've put our best effort to find all vulnerabilities in the system, however our findings shouldn't be considered as a complete list of all existing issues. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

3. Executive Summary

Airdrop is a contract used for distribution of tokens to users and contributors of the Gnosis Safe project.

Safe Token engaged Ackee Blockchain to perform a security review of the Airdrop contract with a total time donation of 3 engineering days in a period between July 13 and July 15, 2022 and the lead auditor was Jan Kalivoda.

The audit has been performed on the commit d997f13.

We began our review by using static analysis tools, namely Slither. Then we took a deep dive into the logic of the contracts. During the review, we paid special attention to:

- ensuring that nobody wouldn't be able to claim any tokens than it is intended,
- detecting possible reentrancies in the code,
- ensuring access controls are not too relaxed or too strict,
- looking for common issues such as data validation.

Our review resulted in 1 medium severity finding.

The contract is very well written and each line has its purpose. The project is nicely readable, well documented and has an extensive test suite.

Ackee Blockchain recommends Safe Token:

- address all reported issues.

Update August 25, 2022: Team provided an updated codebase that addresses issues from this report. See [Appendix C](#) for a detailed discussion of the exact status of each issue.

4. System Overview

This section contains an outline of the audited contracts. Note that this is meant for understandability purposes and does not replace project documentation.

4.1. Contracts

Contracts we find important for better understanding are described in the following section.

VestingPool

The base contract for Airdrop. The VestingPool contract is used to distribute tokens over time to different accounts. Pool manager can use the pool to create vestings (i.e., dedicate a specific amount of tokens to a specified address to be distributed over a specified time) and pause, unpause and cancel vestings that are controlled by the manager (where managed == true).

Airdrop

The Airdrop contract inherits from VestingPool and uses a Merkle proof to create the vestings. Vestings are created with the `redeem` function. Tokens can be claimed with `claimVestedTokensViaModule` or `claimVestedTokens` (inherited from VestingPool). When the airdrop pass the redeem deadline, Pool manager can claim all deposited tokens and the lifecycle of the airdrop contract ends.

4.2. Actors

This part describes actors of the system, their roles, and permissions.

Pool manager

Pool manager can withdraw all tokens after the end of the airdrop.

4.3. Trust model

Unlike VestingPool, vestings created by this contract are not managed by Pool manager and thus can be considered trustless. The only possible identified risk is a call to Pool manager in the `claimVestedTokensViaModule` function, but since users can just earn tokens (if they provide valid proof and other parameters for vesting) it is not a risk at all.

5. Vulnerabilities risk methodology

A *Severity* rating of each finding is determined as a synthesis of two sub-ratings: *Impact* and *Likelihood*. It ranges from *Informational* to *Critical*.

If we have found a scenario in which an issue is exploitable, it will be assigned an impact rating of *High*, *Medium*, or *Low*, based on the direness of the consequences it has on the system. If we haven't found a way, or the issue is only exploitable given a change in configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.) or given a change in the codebase, then it will be assigned an impact rating of *Warning* or *Info*.

Low to *High* impact issues also have a *Likelihood* which measures the probability of exploitability during runtime.

5.1. Finding classification

The full definitions are as follows:

Severity

Severity	Impact	Likelihood
Informational	Informational	N/A
Warning	Warning	N/A
Low	Low	Low
Medium	Low	Medium
Medium	Low	High
Medium	Medium	Medium
High	Medium	High
Medium	High	Low

Severity	Impact	Likelihood
High	High	Medium
Critical	High	High

Table 1. Severity of findings

Impact

High

Code that activates the issue will lead to undefined or catastrophic consequences for the system.

Medium

Code that activates the issue will result in consequences of serious substance.

Low

Code that activates the issue will have outcomes on the system that are either recoverable or don't jeopardize its regular functioning.

Warning

The issue cannot be exploited given the current code and/or configuration (such as deployment scripts, compiler configuration, use of multi-signature wallets for owners, etc.), but could be a security vulnerability if these were to change slightly. If we haven't found a way to exploit the issue given the time constraints, it might be marked as "Warning" or higher, based on our best estimate of whether it is currently exploitable.

Info

The issue is on the border-line between code quality and security. Examples include insufficient logging for critical operations. Another example is that the issue would be security-related if code or configuration (see above) was to change.

Likelihood

High

The issue is exploitable by virtually anyone under virtually any circumstance.

Medium

Exploiting the issue currently requires non-trivial preconditions.

Low

Exploiting the issue requires strict preconditions.

6. Findings

This section contains the list of discovered findings. Unless overriden for purposes of readability, each finding contains:

- a *Description*,
- an *Exploit scenario*, and
- a *Recommendation*

Many times, there might be multiple ways to solve or alleviate the issue, with varying requirements in terms of the necessary changes to the codebase. In that case, we will try to enumerate them all, making clear which solve the underlying issue better (albeit possibly only with architectural changes) than others.

Summary of Findings

	Severity	Impact	Likelihood
M1: The variable `redeemDeadline` can be set to the past	Medium	High	Low

Table 2. Table of Findings

M1: The variable `redeemDeadline` can be set to the past

Medium severity issue

Impact:	High	Likelihood:	Low
Target:	Airdrop.sol	Type:	Data validation

Listing 1. Excerpt from Airdrop.constructor

```
    constructor(
        address _token,
        address _manager,
        uint64 _redeemDeadline
    ) VestingPool(_token, _manager) {
        redeemDeadline = _redeemDeadline;
    }
```

Description

The variable `redeemDeadline` is not sufficiently validated (see Listing 1). It can be set to the past and thus the contract will become useless.

Exploit scenario

By mistake, `redeemDeadline` is set to the past and it is left unnoticed until someone calls `redeem`. As a result, developers need to deploy a new contract.

Recommendation

Add a simple time validation to prevent setting the deadline to the past.

```
require(block.timestamp < _redeemDeadline, "Deadline can not be set to the
past");
```

Go back to Findings Summary

Appendix A: How to cite

Please cite this document as:

Ackee Blockchain, Safe Token: Airdrop, August 26, 2022.

Appendix B: Glossary of terms

The following terms might be used throughout the document:

Superclass/Ancestor of C

A contract that C inherits/derives from.

Subclass/Child of C

A contract that inherits/derives from C.

Syntactic contract

A Solidity contract. May have an inheritance chain, and may be deployed.

Deployed contract

An EVM account with non-zero code. If its source was written in Solidity, it was created through at least one syntactic contract. If that contract had superclasses (parents), it would be composed of multiple syntactic contracts.

Init/initialization function

A non-constructor function that serves as an initializer. Often used in upgradeable contracts.

External entrypoint

A `public` or `external` function.

Public/Publicly-accessible function/entrypoint

An `external` or `public` function that can be successfully executed by any network account.

Mutating function

A non-`view` and non-`pure` function.

7. Appendix C: Fix Review

On August 24, 2022, Ackee Blockchain reviewed Safe Token's fixes for the issues identified in this report. The following table summarizes the fix review. The updated commit was c10da49.

Fix log

Id	Severity	Impact	Likelihood	Status
M1F: The variable `redeemDeadline` can be set to the past	Medium	High	Low	**Fixed**

Table 3. Table of fixes

M1F: The variable `redeemDeadline` can be set to the past

Medium severity issue

Impact:	High	Likelihood:	Low
Target:	M1: The variable `redeemDeadline` can be set to the past	Type:	Data validation

Description

The issue is fixed by adding `require` statement into the constructor that enforces `redeemDeadline` to be set as the future date.

```
require(_redeemDeadline > block.timestamp, "Redeem deadline should be in
the future");
```

Go back to Fix log

ackee

Thank You

Ackee Blockchain a.s.

Prague, Czech Republic

hello@ackeeblockchain.com

https://discord.gg/z4KDUbuPxq