Cookie3 Farming Security Review

Duration: November 13, 2024 - November 19, 2024

Date: February 7, 2025

Conducted by: **KeySecurity**
gkrastenov, Lead Security Researcher

Table of Contents

1 About KeySecurity

KeySecurity is a new, innovative Web3 security company that hires top-talented security researchers for your project. We have conducted over 30 security reviews for various projects, collectively holding over $300,000,000 in TVL. For security audit inquiries, you can reach out to us on Twitter/X or Telegram @gkrastenov or check our previous work here.

2 About AI Agent Layer

Cookie3 utilizes off- and on-chain analytics and an AI data layer to determine quality users who bring value on-chain and reward them for their contribution.

3 Disclaimer

Audits are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

4 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

4.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

4.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

4.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

5 Executive summary

Overview

Project Name	Cookie3 Farming
Repository	https://github.com/Cookie3-dev/farming-contracts
Methods	Manual review

Scope

AirdropClaim.sol
Farm.sol
FarmFactory.sol

Timeline

November 13, 2024	Audit kick-off
November 19, 2024	Preliminary report
November 26, 2024	Mitigation review

Issues Found

Severity	**Count**
High	0
Medium	0
Low	1
Information	6
Total	**7**

6 Findings

6.1 Low

6.1.1 Prevent the overwriting of existing Farm and Airdrop

Severity: *Low*

Context: AirdropClaim.sol#L94

Description: When a new airdrop is created, it is never checked whether an airdrop with the same name already exists. If an airdrop with the same name as an existing one is added, it will overwrite the information of the current airdrop.

A similar problem exists in the `FarmFactory` contract, where new `Farm` contracts are created. It is never checked whether a farm with the given ID has already been created and registered.

Recommendation: Always check whether an airdrop or farm contract with the same name or ID already exists to prevent overwriting.

Resolution and Client comment: Acknowledged.

6.2 Information

6.2.1 Typo in fuction

Severity: *Information*

Context: Farm.sol#L256

Description: In the Farm contract, the function `getCureentTier` has a typo; the name should be `getCurrentTier`.

Recommendation: Change the name of the function to `getCurrentTier`.

Resolution and Client comment: Resolved.

6.2.2 Emit an event in crucial places

Severity: *Information*

Context: Farm.sol#L243

Description: Emit an event in crucial places, such as in the `setRoot()` function in the Farm contract, where the root is updated from the owner.

```
function setRoot(bytes32 _root) external onlyOwner {
        //@audit-issue emit event
        farmingConf.merkleRoot = _root;
    }
```

Recommendation: Emit an event in the `setRoot()` function, following a similar approach to the AirdropClaim contract when the root is updated.

Resolution and Client comment: Resolved.

6.2.3 Redundant errors

Severity: *Information*

Context: Farm.sol#L84-L85

Description: In the Farm contract, the errors `PoolExpired()` and `PoolNotActive()` are never used.

Recommendation: Remove the redundant errors.

Resolution and Client comment: Resolved.

6.2.4 Use custom errors

Severity: *Information*

Context: AirdropClaim.sol#L284

Description: Use a custom error in the `exec` function of the `AirdropClaim` contract. Follow a similar approach as in the entire codebase, where the `require` statement is used and a custom error is returned.

Recommendation: Use custom error in the `exec` function.

Resolution and Client comment: Resolved.

6.2.5 Use safeTransfer instead of transfer for ERC20 tokens

Severity: *Information*

Context: AirdropClaim.sol#L171

Description: In the `AirdropClaim` contract, where the `transfer` function is used, the return parameter is not handled. The `SafeERC20` library is used for `IERC20` to safely handle every transfer operation.

Currently, the `safeTransferFrom` function is only used when the airdrop is added, where the return parameter is checked. However, when a user tries to claim their airdrop, the `transfer` function is used directly instead of `safeTransfer` from the `SafeERC20` library.

Not using `safeTransfer` may cause sweep to fail for some tokens.

Recommendation: Use `safeTransfer` instead of `transfer`.

Resolution and Client comment: Acknowledged.

6.2.6 Missing comments in several places

Severity: *Information*

Context: AirdropClaim.sol#L46

Description: For the error `AirdropDoesNotExist`, the `userClaims` mapping and the constant variables `DIVISOR` and `AIRDROP_CREATOR` are missing comments. The whole codebase is very well documented but comments are missing in these places.

Recommendation: Add comments to ensure the codebase remains well-documented.

Resolution and Client comment: Acknowledged.