AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

TABLE OF CONTENTS

contact@auditmos.com

AUDITMOS

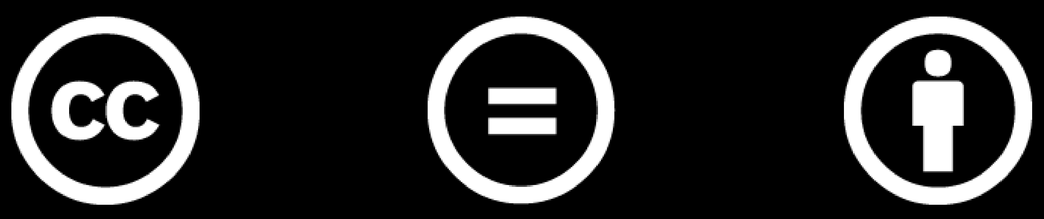
Secure your space

COOKIE3

audit / code review report

January 08, 2025

LICENSE



contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

DISCLAIMER

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

APPROACH AND METHODOLOGY

PURPOSE

1. Determine the correct operation of the protocol, according to the design specification.
2. Identify possible vulnerabilities that could be exploited by an attacker.
3. Detect errors in the smart contract that could lead to unexpected behavior.
4. Analyze whether best practices were followed during development.
5. Make recommendations to improve security and code readability.

CODEBASE

GitHub	https://github.com/Cookie3-dev/
Branch	master
Contracts	AirdropClaim.sol; FarmFactory.sol; Farm.sol; CookieLock.sol

METHODOLOGY

1. Reading the available documentation and understanding the code.
2. Doing automated code analysis and reviewing dependencies.
3. Checking manually source code line by line for security vulnerabilities.
4. Following guildlines and recommendations.
5. Preparing this report.

AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

DESCRIPTION

Issues Categories:

Severity	Description
CRITICAL	vulnerability that can lead to loss of funds, failure to recover blocked funds, or catastrophic denial of service.
HIGH	vulnerability that can lead to incorrect contract state or unpredictable operation of the contract.
MEDIUM	failure to adhere to best practices, incorrect usage of primitives, without major impact on security.
LOW	recommendations or potential optimizations which can lead to better user experience or readability.

Each issue can be in the following state:

State	Description
PENDING	still waiting for resolving
ACKNOWLEDGED	know but not planned to resolve for some reasons
RESOLVED	fixed and deployed

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

AUDIT SCOPE

1. getting to know the project	✓
2. research into architecture	✓
3. manual code read	✓
4. permissions of state changing functions	✓
5. identify common Solidity vulnerabilities	✓
6. test coverage	✓
7. static analysis	✓
8. storage key overlaps	✓
9. DOS possibilities by malicious attacker	✓
10. steal funds possibilities	✓

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

FINDINGS

Finding	Severity	Status
#1 - Users can directly claim locked tokens	MEDIUM	RESOLVED
#2 - Removing a tier will block the claiming functionality	LOW	RESOLVED
#3 - Prevent the overwriting of existing Farm and Airdrop	LOW	RESOLVED
#4 - Emit an event in crucial places	LOW	RESOLVED
#5 - Modifying the tokensForTier of a tier can cause problems	LOW	RESOLVED
#6 - Use safeTransfer Instead of transfer for ERC20 tokens	LOW	RESOLVED
#7 - Use properly name for storage variables	LOW	RESOLVED

contact@auditmos.com

AUDITMOS

Secure your space

COOKIE3

audit / code review report

January 08, 2025

Severity	Status
MEDIUM	RESOLVED

#1 - USERS CAN DIRECTLY CLAIM LOCKED TOKENS

When a user locks tokens, in the mapping userLock only the tier in which the user locked tokens will be stored, as the unlockTime will be set to 0.

During the claiming process, the only requirements the user must meet are: they have staked tokens (i.e., the tier != 0) and block.timestamp > userLocks[msg.sender].unlockTime.

Here, userLocks[msg.sender].unlockTime will always be 0 until the unlock function is called. This means a user can lock their tokens and, when they decide to claim them, they do not need to wait for the unlocking period. Users can directly claim locked tokens without waiting for the unlocking time.

RECOMMENDATION

Users should not be able to claim tokens before the unlocking period has passed.

contact@auditmos.com

COOKIE3

audit / code review report

January 08, 2025

Severity	Status
LOW	RESOLVED

#2 - REMOVING A TIER WILL BLOCK THE CLAIMING FUNCTIONALITY

When the MANAGER decides to remove an existing tier, the whole information about the tier is removed. If, at this moment, a user has an active lock and decides to claim their tokens after the unlocking period has passed, he will receive 0 tokens and will lose all of his staked tokens.

RECOMMENDATION

Do not remove the existing tier. Only prevent users from staking tokens in this tier anymore.

contact@auditmos.com

COOKIE3

audit / code review report

January 08, 2025

Severity	Status
LOW	RESOLVED

#3 - PREVENT THE OVERWRITING OF EXISTING FARM AND AIRDROP

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.

COOKIE3

audit / code review report

January 08, 2025

#4 - EMIT AN EVENT IN CRUCIAL PLACES

Severity	Status
LOW	RESOLVED

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.

contact@auditmos.com

COOKIE3

audit / code review report

January 08, 2025

Severity	Status
LOW	RESOLVED

#5 - MODIFYING THE TOKENSFORTIER OF A TIER CAN CAUSE PROBLEMS

Modifying the tokensForTier of a tier can cause problems when users lock and claim tokens.

If the tokensForTier is increased after every claim, users will receive more tokens than they initially locked. Additionally, the last users who claim their tokens might not receive them due to insufficient tokens in the contract.

In the case where the tokensForTier is decreased, users who have already locked their tokens will receive fewer tokens when claiming, resulting in a loss. These excess tokens will remain stuck in the contract.

RECOMMENDATION

Do not modify the tokensForTier of a tier.

COOKIE3

audit / code review report

January 08, 2025

#6 - USE SAFETRANSFER INSTEAD OF TRANSFER FOR ERC20 TOKENS

Severity	Status
LOW	RESOLVED

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.

contact@auditmos.com

COOKIE3

audit / code review report

January 08, 2025

#7 - USE PROPERLY NAME FOR STORAGE VARIABLES

Severity	Status
LOW	RESOLVED

The storage variable counter is used and incremented when a new tier is added. The purpose of this variable is to avoid duplication of the tiers. Instead of using this name, use tierId, which is a more proper and understandable name for users and developers to know the purpose of the variable.

RECOMMENDATION

Rename the storage variable from counter to tierId.

contact@auditmos.com

auditmos.com

Secure your space

contact@auditmos.com