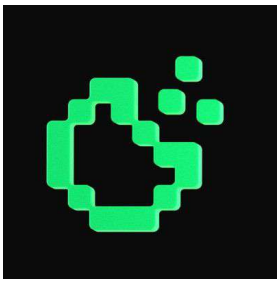

Cookie3 Lock Contracts Security Review

Duration: December 22, 2024 - December 23, 2024

Date: January 10, 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 Cookie3

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 Lock
Repository	https://github.com/Cookie3-dev/cookie-lock
Commit hash	0532bb4236595680fe46cbd1f23fa78f8d34c1bf
Review Commit hash	25dde284f1a240b5ab676fe4c6d6c36ca8dd205f
Documentation	https://github.com/Cookie3-dev/cookie-lock
Methods	Manual review

Scope

CookieLock.sol
CookieLockV2.sol

Timeline

December 22, 2024	Audit kick-off
December 23, 2024	Preliminary report
December 23, 2024	Mitigation review

Issues Found

Severity	**Count**
High	0
Medium	1
Low	1
Information	2
Code Improvement	1
Total	**5**

6 Findings

6.1 Medium

6.1.1 Users can directly claim locked tokens

Severity: *Medium*

Context: Global

Description: 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.

Resolution and Client comment: Resolved. Commit: #25dde284.

6.2 Low

6.2.1 Removing a tier will block the claiming functionality

Severity: *Low*

Context: CookieLock.sol#L143

Description: 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.

Resolution and Client comment: Acknowledged.

6.3 Information

6.3.1 Modifying the tokensForTier of a tier can cause problems

Severity: *Information*

Context: CookieLock.sol#L139

Description: 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.

Resolution and Client comment: Acknowledged.

6.3.2 Redundant errors

Severity: *Information*

Context: CookieLock & CookieLockV2

Description: The following errors and events are redundant and can be removed: `Downgrade` and `CantUpgradeDuringUnlocking` in the `CookieLock` and `CookieLockV2` contracts.

Recommendation: Remove the redundant errors and events.

Resolution and Client comment: Acknowledged.

6.4 Code Improvement

6.4.1 Use properly name for storage variables

Severity: *Information*

Context: CookieLock & CookieLockV2

Description: 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.

Resolution and Client comment: Acknowledged.