Lens V3 Actions Security Review

Pashov Audit Group

Conducted by: Shaka, ubermensch, ast3ros, merlinboii

January 6th 2025 - February 3rd 2025

Contents

1. About Pashov Audit Group

Pashov Audit Group consists of multiple teams of some of the best smart contract security researchers in the space. Having a combined reported security vulnerabilities count of over 1000, the group strives to create the absolute very best audit journey possible - although 100% security can never be guaranteed, we do guarantee the best efforts of our experienced researchers for your blockchain protocol. Check our previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where we try to find as many vulnerabilities as possible. We can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **lens-protocol/lens-v3** repository was done by **Pashov Audit Group**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Lens V3 Actions

Lens is an open social network where every EVM account acts as a Profile, supporting smart wallets and social features. It is built on modular primitives (Feed, Graph, Group, Namespace) that can be extended with custom Actions and Rules, enabling flexible and customizable interactions within the ecosystem.

This scope includes Lens Actions. They serve as examples of how developers could build their own Actions.

5. Risk Classification

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

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

review commit hash - faf3765db6727e5e5e5f5cdd03430870a3cfb0cd

fixes review commit hash - ec11f802a3c8168219dc3725c798562726ef6b70

Scope

The following smart contracts were in scope of the audit:

- `BaseAccountAction`
- `TippingAccountAction`
- `BaseAction`
- `BasePostAction`
- `IERC7572`
- `ISimpleCollectAction`
- `LensCollectedPost`
- `SimpleCollectAction`

7. Executive Summary

Over the course of the security review, Shaka, ubermensch, ast3ros, merlinboii engaged with Avara to review Lens V3 Actions. In this period of time a total of **2** issues were uncovered.

Protocol Summary

Protocol Name	Lens V3 Actions
Repository	https://github.com/lens-protocol/lens-v3
Date	January 6th 2025 - February 3rd 2025
Protocol Type	Social Network

Findings Count

Severity	**Amount**
Medium	1
Low	1
Total Findings	**2**

Summary of Findings

ID	Title	Severity	Status
[M-01]	Collect limit in SimpleCollectAction is incorrectly checked	Medium	Resolved
[L-01]	Creating an immutable collection that has been disabled	Low	Resolved

8. Findings

8.1. Medium Findings

[M-01] Collect limit in `SimpleCollectAction` is incorrectly checked

Severity

Impact: Medium

Likelihood: Medium

Description

On a collect execution in `SimpleCollectAction` it is checked that the collect limit has not been reached.

```
function _validateCollect(
(...)
        if
          (data.collectLimit != 0 && data.currentCollects + 1 > data.collectLimit) {
            revert Errors.LimitReached();
        }
```

However, the check is incorrect, as `data.currentCollects` has already been incremented before the check.

```
uint256 tokenId = ++storedData.currentCollects;

        _validateCollect(originalMsgSender, feed, postId, expectedParams);
```

This means that the transaction can revert even if the limit has not been reached.

Consider the following scenario:

- collectLimit = 1
- Alice collects the post and the currentCollects is incremented to 1
- The validation takes place with the following values `if (1 != 0 && 1 + 1 > 1)` and the transaction reverts

Recommendations

```
-       if
- (data.collectLimit != 0 && data.currentCollects + 1 > data.collectLimit) {
+       if
+ (data.collectLimit != 0 && data.currentCollects > data.collectLimit) {
            revert("Collect limit exceeded");
        }
```

8.2. Low Findings

[L-01] Creating an immutable collection that has been disabled

`SimpleCollectAction` allows disabling the collect action data before the collection contract is deployed.

```
// We don't check for existence of collect before disabling, because it
        // might be useful to disable it initially

        // require(storedData.collectionAddress != address
        //(0), Errors.DoesNotExist());

        require(!storedData.isImmutable, Errors.Immutable());
```

If the collection is created as immutable and was initially disabled, the collection will be unusable, as immutable collections cannot be enabled.

Consider checking if the collection is disabled when it is created as immutable to prevent it from turning unusable.