CANTINA

Balancer v3 post-competition

Security Review

Cantina Managed review by:
Jonah Wu, Lead Security Researcher
Phaze, Security Researcher

December 12, 2024

Contents

1 Introduction **2**
- 1.1 About Cantina . . . 2
- 1.2 Disclaimer . . . 2
- 1.3 Risk assessment . . . 2
  - 1.3.1 Severity Classification . . . 2

2 Security Review Summary **3**

3 Findings **4**
- 3.1 Informational . . . 4
  - 3.1.1 BasePoolFactory lacks total pool count query function . . . 4
  - 3.1.2 VaultAuxiliary event parameters can be improved for indexing . . . 4
  - 3.1.3 Misleading name for permit nonce increment function . . . 4
  - 3.1.4 ERC-4337 bundled transactions may accidentally trigger roundtrip fees . . . 5

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Balancer is a decentralized automated market maker (AMM) protocol built on Ethereum that represents a flexible building block for programmable liquidity.

From Nov 25th to Nov 29th the Cantina team conducted a review of balancer-v3-post-comp on commit hash 73708b75. The team identified a total of **4** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 0
- Informational: 4

3 Findings

3.1 Informational

3.1.1 BasePoolFactory lacks total pool count query function

Severity: Informational

Context: BasePoolFactory.sol#L58-L74

Description: The `BasePoolFactory` contract provides a paginated `getPools()` function for retrieving registered pools but lacks a public function to query the total number of pools. Off-chain services must guess or use trial-and-error to determine pagination parameters since the `_pools` array length is private.

Recommendation: Consider adding a public function to return the total number of pools:

```
function getPoolCount() external view returns (uint256) {
    return _pools.length;
}
```

Balancer: Addressed by PR 1133.

Cantina Managed: Fixed.

3.1.2 VaultAuxiliary event parameters can be improved for indexing

Severity: Informational

Context: IVaultEvents.sol#L258

Description: The `VaultAuxiliary` event parameters `pool` and `eventKey` are not marked as `indexed`, making off-chain event filtering less efficient. Additionally, `eventKey` is defined as `string` when it could be `bytes32` for better gas optimization and indexing capabilities.

Recommendation: Modify the event definition to optimize for off-chain indexing and gas usage:

```
- event VaultAuxiliary(address pool, string eventKey, bytes eventData);
+ event VaultAuxiliary(address indexed pool, bytes32 indexed eventKey, bytes eventData);
```

Balancer: Addressed by PR 1143. Not sure we want to do this (based on previous discussions), but it's there for review/discussion.

Cantina Managed: Fixed.

3.1.3 Misleading name for permit nonce increment function

Severity: Informational

Context: BalancerPoolToken.sol#L158-L161

Description: The `revokePermit()` function name is misleading as it doesn't actually revoke a permit - it only increments the sender's nonce. A permit could still be valid if it was signed with the incremented nonce value. This could lead to developer confusion about the function's actual behavior.

Recommendation: Consider renaming the function to better reflect its behavior:

```
- /// @notice Increment the sender's nonce to revoke any currently granted (but not yet executed) `permit`.
- function revokePermit() external {
+ /// @notice Increment the sender's nonce, invalidating permits signed with the previous nonce.
+ function incrementNonce() external {
      _useNonce(msg.sender);
  }
```

Balancer: Addressed by PR 1145.

Cantina Managed: Fixed.

3.1.4 ERC-4337 bundled transactions may accidentally trigger roundtrip fees

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The transient storage flag `_addLiquidityCalled` used for roundtrip fee detection persists throughout an entire transaction. When multiple UserOperations are bundled by an ERC-4337 bundler into a single transaction, later operations that remove liquidity might unexpectedly incur roundtrip fees if an earlier operation in the bundle added liquidity to the same pool. This could result in higher fees than expected or transaction failures for unrelated users.

Impact: The impact is low since it requires specific conditions - multiple UserOperations interacting with the same pool in a specific order within a single bundle. However, it could affect user experience by causing unexpected fees or reverts.

Likelihood: The likelihood is low. It requires bundlers to include multiple operations interacting with the same pool, and operations must occur in a specific sequence (add liquidity followed by remove liquidity) to trigger the issue.

Recommendation: A suggestion from the Balancer team is to modify the transient storage implementation to incorporate a session ID that changes whenever a pool is unlocked. The current `AddressToBooleanMappingSlot` could be extended to include the session ID in its mapping, making it a double mapping of `session => address => bool`. This would ensure that add liquidity flags are isolated between unlock sessions rather than persisting across an entire transaction.

Alternatively, given the low impact and increased complexity at this stage of the Balancer protocol, this finding can be acknowledged without changes.

Balancer: To us it depends on how we expect this to be used. Will aggregators commonly bundle unrelated transactions together, some of which might involve the same pools? If so, it would make sense to address this.

Cantina Managed: The likelihood of this occurring by chance in the current landscape of things is low. UserOps could also be DoSed on purpose as outlined in the past Cantina-hosted Balancer competition issue "Round-trip fee can be forced on accounts removing liquidity via multisig". Implementing the solution with sessions would align with the intended use of the roundtrip fee and mitigate at least the possibility of triggering the fee by chance. Still, given the low likelihood and increase in complexity, you might want to keep things as they are...