PrimeDAO LBP Contract Audit

Final Report

November 18, 2021

Team Omega

Summary

PrimeDAO has asked Team Omega to audit the contracts that define the behavior of the PrimeDAO LBP contracts.

We found no critical issues.

We classified 1 issue as “medium” - these are issues we believe you should definitely address.

In addition, 8 issues were classified as “low”, and 8 issues were classified as “info” - we believe the code would improve if these issues were addressed as well.

Resolution

An intermediate report was submitted by Team Omega on October 15.
The developers addressed all of the issues in
https://github.com/PrimeDAO/contracts-v2/commit/58da38050d8187f0d3383f63ecbba6a49b4aaace

Scope of the Audit

The audit concerns the contracts committed here:
https://github.com/PrimeDAO/contracts-v2/commit/2399321aa88bb0e241395abfd4666a53ee33f660

And specifically the following contracts and their related tests:

1. LBPManager - contracts/lbp/LBPManager.sol
2. LBPManagerFactory - contracts/lbp/LBPManagerFactory.sol
3. SignerV2 - contracts/utils/SignerV2.sol

We compared the code with the specifications from this document:
https://docs.google.com/document/d/1mTPSRajYhAdX-OcattN3Ai09YHbwvDGwylQPY2sSsT0/

Methods Used

Code Review

We manually inspected the source code to identify potential security flaws.

The contracts were compiled, deployed, and tested in a test environment.

Automatic analysis

We have used several automated analysis tools, including MythX and Remix to detect common potential vulnerabilities. No (true) high severity issues were identified with the automated processes. Some low severity issues, concerning mostly the solidity version setting and functions visibility, were found and we have included them below in the appropriate parts of the report.

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

Findings

General

G1. PrimeDAO copyright is not asserted [low] [resolved]

To quote from the LICENSE file:

```
... attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>
```

Recommendation: Include a copyright attribution if you ever intend to assert your rights
Severity: Low
Resolution: Resolved by adding a copyright statement to each file

G2. OpenZeppelin dependency has a critical vulnerability [low]

Output from npm run audit:

```
@openzeppelin/contracts  4.0.0 - 4.3.1
Severity: critical
UUPSUpgradeable vulnerability in @openzeppelin/contracts - https://github.com/advisories/GHSA-5vp3-v4hc-gx76
TimelockController vulnerability in OpenZeppelin Contracts  - https://github.com/advisories/GHSA-fg47-3c2x-m2wr
fix available via `npm audit fix --force`
Will install @openzeppelin/contracts@4.3.2, which is outside the stated dependency range
```

Although these vulnerabilities do not, a far as we can see, directly affect the code, it is better to use the latest version

G3. Pin dependencies to a specific version in package.json [low] [resolved]

Package dependencies in package.json do not specify a precise version, but rather a range:

```
"@gnosis.pm/safe-contracts": "^1.3.0",
"@openzeppelin/contracts": "^4.1.0",
```

This way of specifying versions may lead to “accidental” upgrades of dependencies by developers that run npm install. This may lead to problems for the dependencies that contain solidity code, such as those mentioned here - when, for example, a trying to verify the deployed bytecode
Recommendation: Use precise versions for contract dependencies
Severity: Low
Resolution: This issue was resolved

G4. Linter emits 25 warnings [info] [resolved]

The linter emits about 25 warnings that pertain to the audited files. Most of these are relatively innocent, although we mention a few of these as separate issues below.
Recommendation: Either fix the warnings, or silence the linter.
Resolution: This issue was resolved

G5. Use latest solidity version [info] [resolved]

The contracts use 0.8.6, the latest version is 0.8.9,
Recommendation: it is best practice to use the latest Soldity version
Resolution: All audited files now use Solidity 0.8.9

G6. Solcover settings file refers to non-existing contract [info] [resolved]

The `.solcover.js` file defining the settings for the coverage refers to a file from the example on the Hardhat docs website. This file doesn’t exist in the project so the reference doesn’t make sense here. Also, there is no configuration to exclude the test contracts from the coverage, which currently damages the coverage total score.
Recommendation: Remove the reference to the non-existing file and instead configure the coverage to skip irrelevant files like the Mock contracts.
Resolution: This issue was resolved

LBPManager.sol

L1. Fee percentages are represented with a wrong precision [medium]

On line 26:

```
        uint256 private constant HUNDRED_PERCENT = 10e18; // Used in calculating the fee.
```

This effectively determines that percentages are defined in units of 1e17 (100% is represented by 1e19). However, the usage seems to suggest that what is really meant here is units of 1e16:

On line 33:

```
        uint256 public feePercentage; // Fee expressed as a % (e.g. 10**18 = 100% fee, toWei('1') =
100%)
```

Not also that this usage is different than the base unit used by balancer for setting the swap fees (where 100% is represented by 1e18)
Recommendation: Avoid confusion and represent the fees with the same precision as Balancer does - i.e. define HUNDRED_PERCENT = 1e18
Severity: Medium - while the code could in theory still work safely with this issue, there is a notable risk that pool creators will configure pools with a “PrimeDAO fee” that is 10 times lower than expected
Resolution: This was resolved according to the recommendation

L2. Missing checks on input variables in initializeLBPManager [low] [resolved]

The function initializeLBPManager is used to configure a pool, which is then deployed by calling initializeLBP. Checks on the correctness of the configuration parameters of the pool are done on creation, and so it is possible to initialize an LBP manager that will throw an error on creation. Examples of such checks are:

- startTime < endTime
- The lengths of _tokenList, _amounts, _startWeights and _endWeights should be 2

Recommendation: Add missing initialization sanity checks
Severity: Low
Resolution: Checks were added

L3. Save some gas by storing result of calculation in memory [low] [resolved]

One lines 193 and 198, _feeAmountRequired() is called - some gas can be saved by storing the calculated result in memory
Recommendation: Store the result of _feeAmountRequired() in memory then use the memory variable instead of calling the function directly
Severity: Low
Resolution: Resolved as recommended

L4. Save some gas by not calling getGradualWeightUpdateParams [low] [resolved]

On lines 232 and 239, getGradualWeightUpdateParams is called to retrieve the end time.
The end time is also available as startTimeEndTime[1], which is cheaper to call
Recommendation: Replace call to getGradualWeightUpdateParams with the locally stored startTimeEndTime[1] variable.
Severity: Low
Resolution: Resolved as recommended

L5. Save gas by skipping the "approve" step [low] [resolved]

On line 245, calling the "approve" function is not needed, as the BPT token already assigns an infinite allowance to the vault:
https://github.com/balancer-labs/balancer-v2-monorepo/blob/master/pkg/pool-utils/contracts/BalancerPoolToken.sol#L58
Recommendation: Remove the unnecessary "approve" call.
Severity: Low
Resolution: Resolved as recommended

L6. Many state variables are possibly not needed [info] [resolved]

This linter warns that the contract uses more than 15 state variables.

```
contracts/lbp/LBPManager.sol
  24:1   warning  Contract has 17 states declarations but allowed no more than 15
max-states-count
```

Recommendation: This warning can be safely ignored. But you could consider slightly changing the logic and to create the LBP in initializeLBPManager rather than only in initializeLBP. This would remove the need to store the LBP configuration (such as tokenList and startWeights) in the LBPManager state before passing it to the LBPFactory. This will save some overall gas costs (although the gas costs for calling initializeLBPManager may be somewhat higher, the cost of calling initializeLBPManager and then initializeLBP will be lower)
Severity: Info
Resolution: The linter warning was silenced

LBPManagerFactory.sol

M1. A floating pragma is set instead of a fixed pragma [low] [resolved]

The Solidity pragma version of this file is set as floating: pragma solidity ^0.8.6 instead of fixed like the rest of the contracts. It is recommended to change this to a fixed one to ensure consistency of the bytecode.
Recommendation: Remove the ^ symbol in the pragma definition to make the solidity version fixed
Severity: Low
Resolution: Resolved as recommended

SignerV2.sol

S1. Missing checks to constructor [info] [resolved]

The approveNewTransaction implementation sensibly checks if the provided contract address and function signature are not 0. The constructor executes no such check.
Recommendation: Require that contract and function signature provided in the constructor are correct
Severity: Info: adding zero values here is harmless
Resolution: Checks were added as recommended

S2. Documentation is outdated [info] [resolved]

One line 20, the documentation from the previous version, Signer.sol, and still mentions the SeedFactory
Recommendation: update the documentation to reflect to new, extended, functionality
Severity: Info
Resolution: Resolved

S3. Error messages refer to Signer instead of SignerV2 [info] [not resolved]

The error messages in the contract are of the form "Signer: can only …" - to be consistent with the rest of the code base, the expected form here is "SignerV2: can only…"
Recommendation: Update the require error messages across the contract to start with "SignerV2" instead of "Signer"
Severity: Info
Resolution: Error messages were left unchanged in order to retain an interface that is consistent with Signer.sol

S4. Consider adding the option to remove approved transactions [info] [resolved]

The contract defines a function approveNewTransaction which extends the list of contract/function signature pairs that can be called. One would expect a corresponding "removeApprovedTransaction" function that allows administrators to remove access to certain functionality.

Recommendation: Add a remove approved transaction function corresponding to the approveNewTransaction function
Severity: Info
Resolution: A function removeAllowedTransaction() was added

Severity definitions

Critical	Vulnerabilities that can lead to loss of assets or data manipulations.
Medium	Vulnerabilities that are essential to fix, but that do not lead to assets loss or data manipulations
Low	Issues that do not represent direct exploit, such as poor implementations, deviations from best practice, high gas costs, etc
Info	Matters of opinion