DeXe Platform TokenAllocator And PoolFactory Update Code Audit And Verification by Ambisafe Inc.

August 2024

Oleksii Matiiasevych

1. **Introduction.** DeXe Network requested Ambisafe to perform a code audit of the DeXe Platform TokenAllocator contract along with the updates to their PoolFactory and other minor changes. The code in question can be identified by the following git commit hash diff:

```
80918ad4fcd23f6b6620b555df4bb30190218bd5
...
5629eb17a37cb6acd6b8ceabdeac6c830aedba2c
```

   All changes are in scope.

   After the initial code audit, DeXe Network team applied a number of updates which can be identified by the following git commit hash:

```
c0291bcc1fbf0db794f3299da5887e08446e753c
```

   Additional verification was performed after that.

2. **Disclaimer.** The code 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 for any specific purpose, or their bugfree status. The code audit documentation below is for internal management discussion purposes only and should not be used or relied upon by external parties without the express written consent of Ambisafe.

3. **Executive Summary.** There are **no** known compiler bugs for the specified compiler version (0.8.20), that might affect the contracts' logic. There were 0 critical, 0 major, 0 minor, 3 informational and optimizational findings identified in the initial version of the

contracts. Most of the findings were addressed and were not found in the final version of the code, they are listed below for historical purposes.

4. **Critical Bugs and Vulnerabilities.** No critical issues were identified.

5. **Line By Line Review. Fixed Findings.**

   5.1. TokenAllocator, line 110. Note, the **closeAllocation()** function doesn't zero out the **allocationInfos.balance** after transferring all the balance to the **allocator**.

   5.2. TokenAllocator, line 211. Optimization, the **_createAllocation()** function reads **lastAllocationId** from storage twice.

6. **Line By Line Review. Acknowledged Findings.**

   6.1. TokenAllocator, line 111. Note, the **closeAllocation()** function doesn't clear **_tokensByAllocator** and **_allocatorsByToken** in case of a last allocation being closed for the pair.

[illegible]

Oleksii Matiiasevych