gra.fun Smart Contracts Review by Ambisafe Inc.
January 2025
Oleksii Matiiasevych

1. **Introduction.** GraFun requested Ambisafe to perform a review of the contracts implementing their token launch platform. The contracts in question can be identified by the following git commit hash:

   `2f86394d6ca8f3294046228a9dd8c96629186eeb`

   There are 9 contracts/libraries in scope.

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

   `aca6300658af7653136f0f9606151b49a4585497`

   Additional verification was performed after that.

2. **Disclaimer.** The review 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.

3. **Executive Summary.** There are no known compiler bugs for the specified compiler version (0.8.28), that might affect the contracts' logic. There were 0 critical, 0 major, 0 minor, 8 informational and optimizational findings identified in the initial version of the contracts. Most of the findings were scheduled to be fixed in the subsequent releases due to their non-essential nature.

4. **Critical Bugs and Vulnerabilities.** No critical bugs or vulnerabilities were found.

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

   5.1. UsingMultipleLiquidityProtectionServices, line 35. Note, the **LiquidityProtection_setLiquidityProtectionService()** function can override the

old pool address.

6. Line By Line Review. Remaining Findings.

6.1. TokenDeployer, line 13. Note, the **deployToken()** function uses **tx.origin** as part of **salt** which could result in a failed deployment in case of meta tx usage.

6.2. TokenDeployer, line 23. Note, the **deployProtectedToken()** function uses **tx.origin** as part of **salt** which could result in a failed deployment in case of meta tx usage.

6.3. TokenFabric, line 25. Note, the **TokenFabric** contract inherits non upgradable version of **ReentrancyGuard** contract, leaving it uninitialized which makes the first call to **nonReentrant** more expensive than subsequent ones.

6.4. TokenFabric, line 183. Note, the **migrateToPool()** function relies upon and sets **isMigrated** value after a potentially unsafe call to **ISwapConnector.addLiquidity()**. Even though it is protected by a **nonReentrant** modifier, it is recommended to always update the sensitive state before making external calls.

6.5. UsingMultipleLiquidityProtectionServices, line 11. Note, the **ProtectionData** struct is not used.

6.6. PancakeV3Connector, line 81. Note, the **addLiquidity()** function has a typo in the price prediction error message, 'inorrect' should be 'Incorrect'.

6.7. ITokenFabric, line 20. Note, the **isMigrated** property of **LiquidityState** struct should be a property of the **TokenInfo** instead, because migration happens only once for all pools simultaneously.

Oleksii Matiiasevych