Disclaimer

An audit isn't a legal document that verifies that the code is secure. Nobody can 100% assure that the code won't have future bugs or vulnerabilities. It's a guarantee that your code has been revised by an expert and it's secure.

Combine.Finance Audit

Combine.Finance is a yield farming platform that allows Uniswap (https://uniswap.org/) liquidity providers to stake their LP tokens and gain COMB token as a reward.

The code is based on the fork of sushiswap (`commit: 2818564c920d26f867ac9c177391010d5f867e22`), with a modification limiting the total COMB supply to 10,000 and a reward decrease mechanism (0.1% per-day). Reward on the first block will be 40 COMB.
Also, the dev address rewards (commission) has been removed.

Audited Contracts

The projects consist of only 2 custom smart contracts (CombToken and MasterChef) and some OpenZeppelin libraries which we decided to skip during the audit, since OpenZeppelin contracts have been audited many times before.

CombToken.sol

Lines of code: 162
Solc version: 0.6.12
Dependencies: SafeMath, Ownable

MasterChef.sol

Lines of code: 195
Solc version: 0.6.12
Dependencies: SafeMath, Ownable, EnumerableSet, SafeERC20, IERC20, CombToken

Findings

Critical Severity

None.

High Severity

None

Medium Severity

1) migrate() is dependent on a currently unspecified migrator contract

Contract(s) affected: MasterChef.sol

Migrator can be set to any contract, potentially allowing the theft of funds, particularly if the owner's private key is compromised. The migrator can be set at any time (and multiple times) by the owner. In the worst case, this could be set to a contract that transfers all underlying LP tokens to an arbitrary address.
Note: since the owner is a timelock contract, users would have 2 days to mass exit the pool before the change could occur.

Low Severity

2) _moveDelegates() may not behave correctly after token transfers

Contract(s) affected: CombToken.sol

The function _delegate() invokes _moveDelegates() with the delegator's full balance instead of remaining undelegated balance. This can cause users to lose delegation ability if additional COMB tokens are acquired without minting (i.e., via transfers).

Consider the following scenario:

1. Alice has 10 COMB, which are delegated to Bob.
2. Alice acquires 1 additional SUSHI from a transfer.
3. If Alice attempts to re-delegate her 11 COMB tokens to Carol, it will fail due to the SafeMath check on L201; effectively, the function will attempt to undelegate 11 tokens from Bob instead of 10, and revert.

In general, if Alice's balance is ever more than the number of tokens minted toward her account (due to transfers), she will not be able to delegate. This can be mitigated by Alice by simply transferring the excess tokens out of her account, however, this scenario may not be clear to end-users from a UX-perspective.

Recommendation

It is not clear if this functionality is as intended. If so, no changes are needed, but user documentation should exist describing the scenario above. If the scenario above is undesirable, _moveDelegates() should be invoked in _transfer() as well. Note however that with this approach, votes can be more easily "bought" by acquiring COMB tokens on exchanges.

3) massUpdatePools() may run out-of-gas if too many tokens are added

Contract(s) affected: MasterChef.sol

This may have implications on calling functions add() and set().

4) Daily reward decrease is ~0.1% but not exactly 0.1%

Contract(s) affected: MasterChef.sol

Calculations have been simplified to avoid high gas usage, so their daily decrease percent may vary, if this as expected behavior, then it should be documented for the community.

Conclusion

No critical or high severity issues were found. Some changes were proposed to follow best practices and reduce the potential attack surface.

Summary

Overall, we are happy with the security posture of the team and the health of the codebase. A lot of fixes and corrections have been done during our collaboration in order to secure and optimize the codebase. We have some reservations about the current architecture but are glad to find the team has considered the implications of their threat model with an intention to upgrade the design where appropriate.