G0 Group

Security Review of

Popcorn

June 2022

Popcorn / June 2022

Files in scope

Following solidity files:

https://github.com/popcorndao/sweet-caramel/tree/6754c218fa8728d936cbc723903cd77693c47f77/packages/hardhat/contracts/core/defi/vault

- AffiliateToken.sol
- Vault.sol
- VaultFeeController.sol

Current status

All discovered issues have been fixed by the developer. There are no known issues in the relevant contracts in https://github.com/popcorndao/sweet-caramel/tree/bafa6fc66a2dda1d653eb8d120d44eddee37d779/packages/hardhat/contracts/core/defi/vault

Issues

1. Anybody can withdraw and redeem on behalf of any user with non-zero allowance

type: security / severity: critical

In `Vault.withdraw` and `Vault.redeem` it's possible to withdraw from allowance, but the functions don't work analogously to `transferFrom` function as could be expected. In `transferFrom` an allowance holder can use the allowance to transfer tokens from the allowance provider to arbitrary address. In withdraw and redeem anybody can call these functions to withdraw shares based on existing allowance between two addresses. This is especially problematic because of the default max approval the vault contract sets to the staking address, this means shares owned by the vault contract can be withdrawn to the staking address at any time by anybody, leading to the loss of these funds.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/popcorndao/sweet-caramel/tree/bafa6fc66a2dda1d653eb8d120d44eddee37d779/packages/hardhat/contracts/core/defi/vault

2. Vault.withdrawAccruedFees doesn't consider pending fees

type: incorrect implementation / severity: medium

`withdrawAccruedFees` ignores pending fees, it neither processes these fees before withdrawal nor considers them when calculating value of the shares being withdrawed.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/popcorndao/sweet-caramel/tree/bafa6fc66a2dda1d653eb8d120d44eddee37d779/packages/hardhat/contracts/core/defi/vault

3. Constructor tries to make an external call to address(this)

type: incorrect implementation / severity: medium

In constructor of `Vault` contract, there's an external call to `ERC20(address(this)).approve()` function when `staking` contract is set. This external call is impossible, because at the time constructor code is being executed, no code is present in `address(this)`.

status - fixed

Issue has been fixed and is no longer present in

https://github.com/popcorndao/sweet-caramel/tree/bafa6fc66a2dda1d653eb8d120d44eddee37d779/packages/hardhat/contracts/core/defi/vault

Notes

- It should be ensured that `withdrawAccruedFees` can't be called too often to avoid the incentive system being abused. The developer stated this will be ensured in the `KeeperIncentive` contract that's still under development.
- Separately stored `redeemFee` can be replaced by calculating it from withdrawal fee, since this equation must hold: `withdrawalFee == redeemFee/(1+redeemFee)`

Following issues have been discovered in the `BaseWrapper` contract that has been developed by a third party. Considering the issues are relatively minor and the contract has been audited by multiple reputable auditing companies and is time-tested, retaining the original code despite these issues is an acceptable course of action.

- In `BaseWrapper` when `estimatedShares` is rounded down to `0` on `line 221` all available shares will be withdrawn on `line 230` instead of withdrawing `1` share, which would be sufficient. This behavior is documented in a comment, but the implementation decision is not explained.
- In `BaseWrapper` on `line 250` it's possible that the `deposit` call will fail due to `depositLimit` in the called `Vault` contract. In that case the withdrawal will fail, this could happen specifically due to previously mentioned issue. The issue is probably unlikely to occur and when it does, user can probably work around it by withdrawing a different amount.