Morpho Blue Convex Wrapper

BROKEN WITH ❤ BY nomoi

The review started on *Monday, December 4, 2023*.

This report was updated on *Monday, March 25, 2024*.

Introduction

Our team has conducted a comprehensive security review of a new Convex Wrapper meant to be used as collateral for Morpho Blue markets. Additionally, we were tasked with reviewing the corresponding oracle to be used for the previously mentioned markets. The primary goal of this assessment was to independently evaluate the security measures, code quality, and overall functionality of the reviewed smart contracts.

The main objective of the reviewed project is to allow holders of Curve LP positions to deposit their positions in the respective Curve gauge through Convex, while still being able to use their positions as collateral in Morpho Blue markets, particularly in the ones described by this Llama Risk proposal.

As always, the Convex team takes an extreme approach to build their systems to be as trustless as possible, having no access to user funds. The codebase was clean and easy to understand, and we were not able to find major issues.

The scope of the review was:

ConvexStakingWrapperMorpho.sol at commit `b95d5b5`

TricryptoLpOracle & TricryptoLpOracleFactory at commit `ba6ae9f`

1. [Client Reported] Factory does not support disabling crvUSD conversion

ENHANCEMENT

After the audit was finished, the client discovered that it was not possible to disable the crvUSD price feed, as it is immutably set upon the factory's construction. Disabling it is necessary for the TryLSD and TriCRV LPs, as the price returned by those contracts is already denominated in crvUSD.

This issue was resolved in commit 22fff24, by making the crvUSD feed's address a function argument that is used when deploying a new oracle through the factory.

2. Atomically call `setMorphoId`

ENHANCEMENT

The `ConvexStakingWrapperMorpho.setMorphoId` function can be called by anyone if `ConvexStakingWrapperMorpho.morphoId` has not been set. The reason for this seems to be that the Convex team expects to deploy the wrapper proxies using the `WrapperFactory` contract, which does not have the functionality to call the `ConvexStakingWrapperMorpho.setMorphoId` function after initialization. If an attacker is monitoring proxy deployments done with the `WrapperFactory` contract, they can call `ConvexStakingWrapperMorpho.setMorphoId` with the wrong id right after the proxy is deployed.

If this attack is performed and detected, the Convex team can just deploy a new proxy and correctly set the id. The only real scenario where this could become an issue is if the wrong id goes undetected and users begin depositing funds, which is a highly unlikely scenario in our opinion.

As an additional safety measure, consider creating a new wrapper factory that calls the `setMorphoId` function right after initialization.

Update: *Acknowledged but won't be fixed since the likelihood for someone to frontrun the* `ConvexStakingWrapperMorpho.setMorphoId` *functions is low.*

3. Handle stale Chainlink prices

ENHANCEMENT

The `TricryptoLpOracle.getPrice` function ignores the timestamp of the last round data when the Chainlink oracle price is queried.

Consider adding some logic to handle stale oracle prices such as including a tolerance threshold and falling back to a different oracle if possible.

***Update**: Acknowledged but since there are no fallback oracles it won't be implemented.*

4. Input Validation

ENHANCEMENT

The `TrycryptoLpOracle` 's constructor calculates and sets the `SCALE_FACTOR` immutable variable. `SCALE_FACTOR` is used to scale the resulting oracle's price (denominated in `crvUSD`) to 18 decimals. For this calculation, it is assumed that the `stETH/ETH` custom price feed being used always uses 8 decimals and the "base" Chainlink Aggregator uses 8 decimals.

However, the previously mentioned assumption is not actually verified, and the decimals are obtained from each aggregator (if provided) when the contract is deployed. If the wrong aggregator addresses are used, (e.g. if one of the base aggregators does not use the correct decimals), the `SCALE_FACTOR` calculation will result in the wrong value and might go unnoticed.

Consider including an additional check to verify that the provided Aggregator addresses use the correct number of decimals.

***Update:** As of commit a8aedbb this issue was fixed by adding validations to the `TrycryptoLpOracle` 's constructor.*

5. Simplify `TricryptoLpOracleFactory.newTricryptoLpOracle` function

ENHANCEMENT

The `TricryptoLpOracleFactory.newTricryptoLpOracle` function can be simplified in order to improve its readability, by making the `TricryptoLpOracle` contract implement the `IOracle` interface, and changing the function to:

```
function newTricryptoLpOracle(
    address _curveTriPool,
    address _baseFeed,
    address _lidoFeed
) external returns (IOracle) {
    return new TricryptoLpOracle(
        _curveTriPool,
        _baseFeed,
        _lidoFeed,
        CRVUSD_FEED
    );
}
```

Update: *As of commit a8aedbb this issue was fixed by making the `TricryptoLpOracle` contract implement the `IOracle` interface and updating the `TricryptoLpOracleFactory.newTricryptoLpOracle` function.*

6. Use contract types for function and constructor arguments

ENHANCEMENT

The `TricryptoLpOracle` 's constructor accepts addresses as parameters, same as the `TricryptoLpOracleFactory.newTricryptoLpOracle` function. Even though contract types are effectively the same as addresses, using types in the arguments can help prevent simple bugs such as passing the arguments in the wrong order.

Update: *Won't fix .*