TWEAG
by Modus Create

Technical Review of Cerra AMM

High Assurance Software Group
March 8, 2024

Contents

Chapter 1

Executive Summary and Scope

This report is presented without warranty or guaranty of any type. This report lists the most salient concerns that have so far become apparent to Tweag after a partial inspection of the engineering work. Corrections, such as the cancellation of incorrectly reported issues, may arise. Therefore Tweag advises against making any business decision or other decision based on this report.

Tweag does not recommend for or against the use of any work or supplier referenced in this report. This report focuses on the technical implementation provided by the project's contractors and subcontractors, based on their information, and is not meant to assess the concept, mathematical validity, or business validity of Cerra's product. This report does not assess the implementation regarding financial viability nor suitability for any purpose.

Scope and Methodology

Tweag looks exclusively at the on-chain validation code provided by Cerra. This excludes all the frontend files and any problems contained therein. Tweag manually inspected the code contained in the respective files and attempted to locate potential problems in one of these categories:

a) Unclear or wrong specifications that might allow for fringe behavior.

b) Implementation that does not abide by its specification.

c) Vulnerabilities an attacker could exploit if the code were deployed as-is, including:

- race conditions or denial-of-service attacks blocking other users from using the contract,
- incorrect dust collection and arithmetic calculations (including due to overflow or underflow),
- incorrect minting, burning, locking, and allocation of tokens,
- authorization issues,

d) General code quality comments and minor issues that are not exploitable.

Where applicable, Tweag will provide a recommendation for addressing the relevant issue.

Chapter 2

Audit

2.1 Methodology

Tweag analyzed the validator scripts comprising Cerra AMM, contained in an archive sent by Cerra. The names of the files considered in this audit and their `sha256sum` are listed in Table 2.1.

`sha256sum`	File Name
`d69e3a4...62e279c`	`Cerra/BatchOrder/OnChain.hs`
`4e895f4...5656527`	`Cerra/BatchOrder/Types.hs`
`0bd5218...9245c0b`	`Cerra/ConstantProductFactory/OnChain.hs`
`8cc49a9...1601823`	`Cerra/ConstantProductFactory/Types.hs`
`523921e...2a70c22`	`Cerra/ConstantProductLiquidity/OnChain.hs`
`433705d...5542042`	`Cerra/ConstantProductPoolNFT/OnChain.hs`
`19ab2c2...4299827`	`Cerra/ConstantProductPoolNFT/Utils.hs`
`89f7c2d...d264a3c`	`Cerra/ConstantProductPool/OnChain.hs`
`273a6ff...d662783`	`Cerra/ConstantProductPool/Types.hs`
`4eee8e6...78f38c2`	`Cerra/ConstantProductPool/Utils.hs`
`e54c31f...f22a013`	`Cerra/Types/Coin.hs`
`9f8699d...b5cb70c`	`Cerra/Utils/Debug.hs`
`42ab33d...2eacc1d`	`Cerra/Utils/OnChainUtils.hs`

TABLE 2.1: *On-chain code source files and their `sha256sum` that were analyzed as part of the review*

Our code review is based on meetings and messages exchanged with Cerra.

2.2 Findings

Table 2.2 lists our concerns with the current Cerra AMM implementation based on our partial exploration during a limited period of time. Throughout the rest of this section, we will detail each of our findings.

2.2.0.1 ■ No fairness is guaranteed nor encouraged when applying orders

Severity: Low

Orders are applied in a batch. That is, a specific entity who owns a batcher token is able to execute a certain amont of orders picked among orders present at the batcher script address. However, there are no guarantees on which orders will actually be applied and which ones will be ignored. Furthermore, there

Severity	Section	Summary
High	2.2.0.3	The product lacks a detailed specification
Medium	2.2.0.8	Implementation of equality test for `PoolDatum` is incomplete
Low	2.2.0.1	No fairness is guaranteed nor encouraged when applying orders
Low	2.2.0.2	Checking the batcher token could be done with a reference input
Low	2.2.0.4	The pool NFT and factory token might be merged
Low	2.2.0.5	Usage of map for additional info in `PoolDatum` is inadapted
Low	2.2.0.6	Remnants of the profit sharing feature take up space
Low	2.2.0.7	Precision is a static value that takes up space in pool datums

TABLE 2.2: *Table of findings*

are no constraints on the order in which orders are applied. Considering that each order application will in turn change the rate between the pool assets, this lack of fairness can be harmful to users.

While the current state of the product partially accounts for this lack of fairness by having user express requirements in their order, there are more rigorous ways that a smart contract can ensure fairness of orders. Here are two examples of common fairness implementation in the Cardano world:

- A mechanism of complaints could be implemented, where users would be able to issue complaints if an order that has been issued after theirs is applied before. Users would be rewarded when that happens. This requires to timestamp the orders.
- A mechanism of chained orders could be implemented, where orders, instead of being singletons at a script address, would be part of a chain of order, while the batcher would be forced to execute them in the order they appear in the chain. Once more, a timestamp would be required.

2.2.0.2 Checking the batcher token could be done with a reference input

Severity: Low

When the batcher applies a batch of orders over a pool, the batcher token has to be shown. As it stands now, showing this token is done the old way: the output containing the token is consumed and created again by the transaction.

This feature could be improved by having the token shown in a reference input instead. This would bear the usual benefits of using reference inputs (reducing the size of the transaction), with the addition of standardizing and simplifying the current `ApplyPool` redeemer. Indeed, at the moment the redeemer contains an index which points to the index of the batcher input among the inputs of the transaction. This is not standard as the order of inputs is usually unspecified in a transaction (as opposed to outputs).

2.2.0.3 The product lacks a detailed specification

Severity: High

Cerra AMM is unspecified. The fact that it comes from an existing product with a few adjustments should not mean that it is devoid of any kind of specification. Each transaction and the associated

endpoints should be specified both at high level (what they are trying to accomplish within the product life span) and at low level (what specific actions involving the script and minting policies should happen when executing a certain action).

Without any kind of specification, this is impossible to assess correctness, as correctness can only be checked with respect to an expected behavior. In the specific case of this product, a detailed specification would be in particular extremely beneficial for the application of orders. Here are some of the questions it should answer:

- Is the batcher incentivized to pick some orders instead of others when sending a batch of orders for applications?
- Are the orders in a batch processed one by one (thus each one modifies the equilibrium in the pool) or aggregated before application (thus the batch counts as a single modification of the pool)?
- What exactly is the meaning of each of the fields in the `OrderDatum` and associated steps?

In addition to making any review more difficult, a lack of specification is also detrimental to the development and maintaining team, as it is the only way to ensure a solid understanding of the product itself, as well as to ensure a similar understanding between team members and involved parties overall.

2.2.0.4 The pool NFT and factory token might be merged

Severity: Low

The product contains 3 minting policies. The factory token, the pool NFT and the liquidity tokens. Liquidity tokens are meant to represent the amount of liquidity a user contributed to a pool. The NFT provides a unique id based on the UTxO spent to mint it. The factory token ensures pools are initialized correctly (in terms of value and datum). It may be possible to merge those two into one single token and minting policy that both consumes a UTxO (and keep its hash as a token name) and checks the pool UTxO is well initialized (datum and value).

2.2.0.5 Usage of map for additional info in `PoolDatum` is inadapted

Severity: Low

Cerra AMM introduces an additional `pdPrices` field in pool datums (`PoolDatum`). This field is a map from `String` to `Integer`. In absence of detailed specification, and instead referring to the validators, the elements present in this map will always be key/value pairs for keys “amountA”, “amountB”, “priceA”, “priceB”, “precisionA”, and “precisionB”’. They are always expected to be present as checked by the `validatePrices` helper function of the pool validator.

Therefore, we suggest to include them as dedicated fields in `PoolDatum` instead for simpler and lighter representation and space usage, and also to benefit from higher type safety. Right now, any field name could be omitted or misspelled by mistake.

We understand this price information is supposed to be read as reference inputs. Making those values actual fields of the datum, or making the `pdPrices` field a static record type, would not hinder this use case.

If there are additional optional values to be included, these could be added as optional fields in the datum or the `pdPrices` field. Maps should be used in case keys can be arbitrary, which does not seem to be the case in the product.

2.2.0.6 Remnants of the profit sharing feature take up space

Severity: Low

As confirmed by Cerra, the profit sharing feature is not used. However there are traces of the feature left in the code. For example, there remains a dedicated optional field in the datum. It is even required to be set to `Nothing` by the factory token minting policy. In turn, in the pool validator, local variable `feeOn` will always be `False`, and the evaluation of `deltaLiquidityShare` will never involve the `calculateProfitSharing` function. Nevertheless, the implementation of `calculateProfitSharing` will appear in the compiled pool validator.

This field in the datum, along with `calculateProfitSharing` in the validator, could be removed to further optimize transaction size (datum size and script size).

2.2.0.7 Precision is a static value that takes up space in pool datums

Severity: Low

In `PoolDatum`, the new `pdPrices` field is a map that is required to contain entries for keys "precisionA" and "precisionB". These entries are required to have a value of exactly 12 by the pool validator in `validatePrices`. This value 12 is static and hardcoded in the validator. It is not an actual dynamic state value.

Therefore, requiring it to be inside the datum seems inappropriate. It makes datums and transactions slightly heavier for no added benefit.

We suggest to turn precision into a script parameter of the pool. Which precision is associated to which script address is a piece of information that can be relayed offchain. Another possibility would be to remove this parameter completely and simply always consider a precision of 12 digits.

2.2.0.8 Implementation of equality test for `PoolDatum` is incomplete

Severity: Medium

The additional `pdPrice` field in `PoolDatum` is not taken into account in equality checking. The `Eq PoolDatum` instance omits the map of prices. Although this poses no security risk at the moment, this is likely to be error prone in further iterations of the product.

We suggest to provide a full trustworthy `Eq` instance or at least document the current implementation quirk. We think that applying suggestions described in 2.2.0.5 (using dedicated fields rather than maps) would simplify the implementation of the `Eq` instance.

2.3 Conclusion

This report outlines the 8 concerns that we have gathered while inspecting the design and code of Cerra AMM, pertaining to the code contained in the files listed in Table 2.1. As stated in Chapter 1, Tweag does not recommend for nor against the use of any work referenced in this report. Nevertheless, the existence of a *high* severity concern is a warning sign.