Platonic.Systems

*for Project Ardana*

# Audit

*Authors*
Quinn Dougherty quinn.dougherty@platonic.systems
Foo Bar foo.bar@platonic.systems

*Supervisor*
Isaac Shapira isaac.shapira@platonic.systems

September 29, 2021- 19:24

# Table of Contents

# I. Preamble

## I.1. On the approach

There are a number of ways to go about this.

The first way we talked about was to provide a list of desirable properties and find ways to verify or validate them. This is a sort of *positive approach*, it talks about things we want.

Another angle is for me to write up my understanding of all the ecosystem functionalities one at a time and point out some threatmodels along the way. This is a sort of *negative approach*, it talks about things we don't want.

The problem with both approaches is the distance between *pointing out* properties and *implementing/defending against them* depending on if it's the positive or negative case. Also keep in mind that in some sense the audit is a "warmup" for a post-launch formal verification.

## I.2. Desiderata

- **The community is the audience. The community is the customer.**
- I think we want to make an explicit demarcation of risks for whom. Some risks only effect Ardana internally, other risks effect the collective of Dana holders, other risks effect our neighbors in the ecosystem. I think there's a difference between risks we want to eliminate and risks we want to empower the community to take. I'm sort of imagining profiling these as separate *axes* of risk. The idea here is that I'm frustrated with the literature because it almost feels like it uses the word "attack" anytime someone loses money. This is not adequate- sometimes you just gambled and lost, sometimes the "attacker" followed the rules of the mechanism as designed. There's a morality/values question to each attack, and **the audit needs to provide disambiguation and clarity without taking sides on each values/morality question**.

- The audit is **as much as was possible to do before launch time, not exhaustive**.

We should also clearly define what we think a secure DeX even is.

# II. On datastream integrity

There are a couple places in the Ardana ecosystem that suggest **interaction with live datastreams**. A variety of software concerns such as API integration practices or keeping a stream open are ignored in the current document.

One of these such places is the oracle/bot from the governance layer. The oracle is to consume *third party* price data from something like coinbase, binance, etc. and produce transaction signatures.

The second such place is a proposal Morgan and I discussed on discord for DEX subpool size limits. If we choose to do something functional/dynamic, rather than constant/static, for the subpool size limits the most principled choice is to *infer* them from data. > We can determine it empirically using data such as the data cited here and analytical formulas for determining the pool sizes required to reach slippage targets for different swap sizes (Morgan)

While it's absolutely an option to download a static dataset once (and refresh it every few months or so), the natural question is *do we better serve the project by implementing online learning?* Here I am assuming that somewhere in the literature a formula is written down, making the actual model dead simple. But since the possibility of live datastream has been floated, I want to enumerate some of the pitfalls.

I will provide a few threatmodels that arise when a datastream is interacted with "online".

## II.1. Threatmodel 1: third parties compromised

The idea is *we do not trust the third party data sources*. Attackers here fall into two camps: those who are trying to corrupt the beliefs of the whole market and those who are trying to corrupt our beliefs in particular.

I'm envisioning something like the creation of artificial (even fraudulent) arbitrage opportunities by directly perturbing coinbase/binance's beliefs through hacking, and Ardana's behavior is *downstream* of coinbase/binance's beliefs because of where in our system we interact with their APIs.

Suppose we implement the online learning version of subpool size limit selection. An attacker may be able to arbitrage on pool sizes somehow iff they can force an irrational choice of pool sizes (and if our pool sizes are downstream of coinbase/binance data, all they'd need to do is hack into coinbase/binance).

### II.1.1. Mitigation

Does coinbase/binance have a notification system that goes out to API users when they detect a breach? If so, we should handle it almost as an error and fallback to the last uncorrupted snapshot when such a notification is retrieved.

Test for agreement between multiple sources. The probability that an attacker compromises multiple data sources in exactly the same way is much lower than the probability that an attacker compromises one of them.

## II.2. Threatmodel 2: edge case behavior of model

Even if model is dead simple, it could still go off the rails if it got bizarre, unforeseen inputs.

### II.2.1. Analogy

In a more intricate model, **out-of-distribution robustness** describes the resilience of that model to *shifts in input distribution* or, in the extreme case, an attacker eliciting behavior that the model creator does not want by finding inputs on which the model behaves pathologically. As you can imagine, it is easier for attackers to simply make the model fail (make wrong predictions, for instance) than it is for attackers to target behaviors that they desire (make the model benefit them in some way, for instance).

### II.2.2. Mitigation

Be extremely liberal in property tests, this is not a time to save testing resources from implausible cases, because implausible cases could be what hurts us.

Be extremely conservative in input validation/constraints, though beware a lot of inference/data decisions are being made when such constraints are imposed.

## II.3. Threatmodel 3: positive feedback loop

Optimistically trades and prices made on our DEX system will be a part of the data from coinbase/binance. What does it mean when our behavior shows up in the data from which we derive our behavior?

A potential pitfall here isn't entirely unlike threatmodel 2. Under positive feedback, data can simply be sent off the rails into "edge case behavior" that we didn't think would show up in the operating of our model.

### II.3.1. Analogy

Fraud detection at Stripe is trained on the prior year's fraud data. The problem is that data is labeled by the earlier iteration of the model, so a perturbation in the model's behavior might lead to a (nonlinearly drastic) perturbation in the new data labels.

### II.3.2. Mitigation

Via the Stripe example, we can do something called *counterfactual evaluation* which is an algorithm for generating data "as if" our behavior wasn't influencing the data. If this seems important/promising I can workshop with Bassam what this would look like for us.

# III. Scalar types

We use `Rational` to represent numbers in the contract. As of this writing, tolerance (number of decimals needed to evaluate equality) is set to 30.

For the smart contract, we require calculations which are highly precise and can handle very large numbers and can be reproduced exactly across different hardware. Using FLOPs (floating point operations) is not compatible with these requirements. We are not able to determine exactly how big or how precise our numbers need to be, so we cannot say that FLOPs allow for enough size and precision. We can say, however, that FLOPs are implemented slightly differently on different hardware and results may not be reproducible across different hardware. Additionally, FLOPs are not allowed to be used in Plutus on-chain code. These are the constraints which do not allow us to use `Double` to represent numbers in the smart contract.

# IV. Vampire attack

Wouldn't it be great if there were unique affordances by Cardano itself that prevented vampire attack?

## IV.1. What is it?

According to my research, this attack happened once. Finematics' coverage is excellent, so I'll defer explanation of the base level occurrence to them. But here I will attempt to provide a generalized notion.

# V. Flashloans

Flashloans are associated with something like `$136M` in losses.

Ethereum offers flash loans because they have **multi-step atomic transactions**. Cardano does not have these. So in spite of `Aada`'s claims, we are not expecting flash loans to enter the Cardano ecosystem at this time. There may be lessons from the attacks in the reports, but they are not entirely straightforward. The question is: is there anything *unique* introduced by the flash loan mechanism? The auditing research opportunity here is not a huge priority.

### V.0.1. Action: monitoring Cardano for developments in multistep atomic transactions

Project Ardana will be monitoring the evolution of Cardano, because we believe that if multistep atomic transactions are introduced flashloans will be shortly around the corner.

### V.0.2. In this event, the following mitigation strategy sketches will become urgent

- Onchain code only allow interop from one platform and users, not arbitrary platform.
- Lending products ought to require to collateralize in one whole transaction ahead of time before.
- Block price manipulation by disallowing mid-transaction information from updating prices.