

January 18, 2024

---

## Arbiter Security Review

### ENGAGEMENT II

Waylon Jepsen	Lead
Colin Roberts	Lead
Could be you	Researcher

# 1 Executive Summary

Over the course of X days in total, [Protocolname](#) engaged with [Spearbit](#) to review [Protocolname](#) protocol.

We found a total of X issues with Protocolname.

Repository	Commit
<a href="#">Projectname</a>	<a href="#">commithash</a>

## Summary

Type of Project	TYPE
Timeline	Feb 29, 2022 - Feb 31, 2022
Methods	Manual Review
Documentation	High
Testing Coverage	High

## Total Issues

Critical Risk	1
High Risk	1
Medium Risk	0
Low Risk	0
Gas Optimizations and Informational	0

# Contents

<b>1</b>	<b>Executive Summary</b>	<b>1</b>
<b>2</b>	<b>Primitive</b>	<b>3</b>
<b>3</b>	<b>Introduction</b>	<b>3</b>
3.1	Dispute Game . . . . .	3
3.1.1	How does OpLabs define sucess . . . . .	3
3.1.2	Conservative Time Estimates . . . . .	3
3.1.3	Resources on the dispute game . . . . .	4
3.2	Agent Based Modeling . . . . .	5
3.3	Simulation Components . . . . .	5
3.4	Simulation Setup . . . . .	6
3.5	Risk Modeling . . . . .	6
<b>4</b>	<b>Findings</b>	<b>6</b>
4.1	Critical Risk . . . . .	6
4.2	High Risk . . . . .	6
4.2.1	Issue title (Only first word should be capitalized; titles should never end with punctuation) . . . . .	6
4.3	Medium Risk . . . . .	7
4.4	Low Risk . . . . .	7
4.5	Gas Optimizations . . . . .	7
<b>5</b>	<b>Additional Comments</b>	<b>7</b>
<b>6</b>	<b>Appendix</b>	<b>7</b>

## 2 Primitive

Primitive is a team of deeply technical passionate individuals, building the future of finance. You can find more information about us at [primitive.finance](https://primitive.finance).

## 3 Introduction

This report is a security review of the Optimism Decentralization infrastructure: The dispute game.

### 3.1 Dispute Game

The dispute game is a game theoretic game that will decentralize the technical infrastructure of the Optimism Stack. This engagement is designed to investigate these assumptions at both a game theoretic level and a technical level. The focus of the security review was trying to break the two assumptions

#### 3.1.1 How does OpLabs define success

I asked ben how he would define success for the engagement and he was thinking (very rough estimates we can change) Success:

- 100M Sims run on a (single or many but takes about 45min to compute) cannon trace (on avg 1.5b instructions) with at least 5 agents behaviors.
- Bonding: Something showing the volatility at risk of bonding profile (the probability of a bond going under water for given gas worst case gas movement and what that would mean for Layer one). (was a specific ask)
- A report with analysis insights outlined for decision makers (this document will evolve into this).
- I(Waylon) believe there is value to be offered in theoretical results of strong security properties. This is more the tastful way in which we can deliver this.

#### 3.1.2 Conservative Time Estimates

I believe that if we allocated 2-3 people (Waylon, Colin, maybe Matt), full time work it would take 8 Weeks. I think that we can't work on all of it all the time but there is certainly going to be things that everyone can help out with. That being said I think that Matt and Colin have the most experience writing simulations and are the strongest

candidates for the job. I waylon want to own this engage and take responsibility if things go wrong

### 3.1.3 Resources on the dispute game

- [Optimism Dispute Game Overview](#)
- [Optimism Specs](#)
- [Durin](#)
- [notion on some elementary bond analysis](#)
- Our Shared Repository: Pending clabby on this

A dispute game is an abstraction over a state machine that takes in an input from an ordered set (instruction).

State machine is defined as a five tuple  $\{\Sigma, S, S_0, \delta, F\}$  follows:

$\Sigma$  is the Set of all inputs denoted by an alphabet  $\sigma$ .  $\Sigma$  represents all the possible states of the MIPs Cannon VM (link to cannon), which is a subset of 55 instructions of the MIPs architecture and several linux sys calls. The system calls are for user io which are done through the kernal.

In the dispute game state machine both the states and the inputs are the same such that  $\Sigma = S$ .

The initial State  $S_0$  is the state a constant refered to as the absolute prestate.

The state transition function denoted  $\delta : (S, O) \rightarrow S$  where  $O$  is external data from a system call to a contract asking for the pri-image of a hash (could be any 0-4.8MBs of data(can we compute the size of that search space?))) and returns a new state. In practice, this is implemented by querying a pre-image of a hash Final state is a state  $s$  in  $S$

The two big high level assumptions we are trying to break are:

1. An honest player should never loose the dispute game.
2. It should always be profitable to be an honest player.

Meaning we want to attempt to provide evidence by counter example.

**Game resolution (solving the game)** This is what a *cannon-trace provider* does when it solves for the correct trace. This is also how durin will be solving our game states.

The game state is a binary tree where each node is a state and each leaf is an instruction. The instructions are ordered left to right (first to last) and read in (Ask ben how many at a time).

**scratch notes i need to clean up about game resolution algorithm** We define "position" as follows: a position has an index, which is calculated using the formula  $(2^{\text{self.tree\_depth}} + \text{self.index\_at}(\text{zero indexed from left to right depth}))$ .

In the context of the game, the only actions a player can take are to either agree or disagree to play. Durin functions as a state machine, taking in inputs (r). The way durin's cannon trace solver works is described in the bisection algorithm. The same first move (disagree, or disagree) must always be made in order to disagree with a trace. this is done by a tree bisection algorithm in practice, where the leafs of the nodes are the instructions

*Disclaimer:* This security review does not guarantee against a hack. It is a snapshot in time of brink according to the specific commit by a three person team. Any modifications to the code will require a new security review.

## 3.2 Agent Based Modeling

Arbiter uses agent based modeling with the rust evm to provide security and risk analysis insights that are traditionally more difficult to audit. Our agent architecture for the dispute game is as follows:

- Oracle Agent: Responsible for syncing the dispute game state by loading the latest claim and then solving the correct move for that claim by making an api call to durin. The oracle agent will then send honest moves to the honest agent.
- Honest Agent: The Honest Agent is responsible for receiving the honest moves from the oracle agent and then acting on them in the dispute game.
- Dishonest Agent: The Dishonest Agent is responsible for acting attempting to resolve an incorrect move in the dispute game, We will perturb the dishonest agent to look for insecurities in the protocol.

## 3.3 Simulation Components

The system is composed of several agents and contracts. Below is a summary of the components:

- **Agents:**

- Oracle Agent: Responsible for syncing the dispute game state by loading the latest claim and then solving the correct move for that claim by making an api call to durin. The oracle agent will then send honest moves to the honest agent.
  - Honest Agent: Responsible for receiving the honest moves from the oracle agent and then acting on them in the dispute game.
  - Dishonest Agent: Responsible for attempting to resolve an incorrect move in the dispute game. We will perturb the dishonest agent to look for insecurities in the protocol.
- **Contracts:**
    - Dispute Game: Holds all moves currently in the dispute game.
    - Dispute Game Factory: Has pointers to all created dispute games.
  - **Oracle:**
    - Durin: A single oracle used in the system.

## 3.4 Simulation Setup

The state of the evm we simulated on was created by running a forge deploy script to an anvil instance and then dumping the state. The instance was then loaded into revm through arbiter.

## 3.5 Risk Modeling

We will perturb over the infinite space of dishonest actors for the dispute game. We will also perturb various L1 preposals. This will enable us to model the risk of the protocol in a more robust way. We will also measure the bond mechanics of the dispute game to ensure that there are no game theoretic attacks that can be made on the protocol.

## 4 Findings

### 4.1 Critical Risk

### 4.2 High Risk

#### 4.2.1 Issue title (Only first word should be capitalized; titles should never end with punctuation)

**Severity:** High

**Context:** `Contract.sol#L160-L165`

**Description:**

```
contract Test {  
    ...  
    // Code blocks must be indented with 4 spaces.  
}
```

**Recommendation:**

```
+ use diff syntax to describe what should be changed  
- ...
```

**Project:** Fixed in [PR #1](#).

**Spearbit:** Resolved.

### 4.3 Medium Risk

### 4.4 Low Risk

### 4.5 Gas Optimizations

## 5 Additional Comments

\clearpage

## 6 Appendix