Primitive

Primitive Bits, Inc.
New York, NY
primitive.xyz

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**
Projectname	commithash

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

2 Primitive

Primitive is a team of deeply technical passionate indciduals, building the future of finance. You can find more information about us at 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 engadgement is designed to investigate these assumptions at both a game thoeritic 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 sucess

I asked ben how he would define sucess for the engadgement and he was thinking (very rough estimates we can change) Sucess:

- 100M Sims run on a (single or many but takes about 45min to compute) connon 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), I think we can make suggestions on the best way to price bonds here.
- 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 How Does Primtive Define Sucess

- A report with analysis insights outlined for decision makers (this document will evolve into this).
- By meeting optimisms success criteria.

- Improving our arbiter api and number of arbiter related assets.
- Capturing Brand value

3.1.3 Conservative Time Estimates

I believe that is we allocated 2-3 people (Waylon, Colin, maybe Matt), full time work it would take 8 Weeks (ben was trying to convince me we could do it in a month), its possible we could get it done in less. 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 our the strongest candidates for the job.

3.1.4 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:

Σ is the Set of all inputs denoted by an alphabet σ. Σ 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 *connon-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).

For any node n on the set of nodes in our tree N The position of a node as an index, which is calculated using the formula $f(n) = 2^{n_d} + n_i(Askbenhere)$ where n_d is the depth of the node in the tree and n_i is the index zero indexed from left to right at this 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 bisdection 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 dishost actors for the dispute game. We will also perturb various L1 preposals. This willenable 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