dxDAO Omen Arbitrator Review No. 1

Feb, 2021

Chapter 1

Introduction

1.1 Scope of Work

This code review was prepared by Sunfish Technology, LLC at the request of members of dxDAO, an organization governed by a smart contract on the Ethereum blockchain. The code covered by this review (see section 1.2) implements a contract that allows the dxDAO `Avatar` contract to act as an arbitrator for disputes (and "questions") for a third-party platform.

1.2 Source Files

This review covers code from the following public git repository and commit SHA:

`github.com/nicoelzer/omen-arbitrator`
`c2b532084998a42b5de0da54bb6cf2c0cf8e4fa1`

Within that commit, only the following files were reviewed:

- contracts/DXdaoArbitrator.sol

This review was conducted under the optimistic assumption that all of the supporting software infrastructure necessary for the deployment and operation of the reviewed code works as intended. There may be critical defects in code outside of the scope of this review that could render deployed smart contracts inoperable or exploitable.

1.3 License and Disclaimer of Warranty

This source code review is not an endorsement of the code or its suitability for any legal/regulatory regime, and it is not intended as a definitive or exhaustive list of defects. This document is provided expressly for the benefit of dxDAO developers and only under the following terms:

Chapter 2

Moderate Defects

Issues discussed in this section are code defects that may lead to unintended deviations in behavior. It may be possible to chain multiple moderate defects into a working exploit.

2.1 Missing Re-entrancy Guard

The `requestArbitration()` function can be called by any Ethereum address, and it transfers Ether, manipulates internal state, and calls external contracts. Consequently, this function *could* be exploitable through a re-entrancy attack, or it could enable an attack against one of the contracts that it calls.

Consider using a re-entrancy guard (like OpenZeppelin's `ReentrancyGuard`) to protect the `requestArbitration()` function and its callees.

Chapter 3

Minor Defects

Issues discussed in this sections are subjective code defects that affect readability, reliability, or performance.

3.1 Non-standard Function Naming

The `setOwner()` function should be renamed to `changeOwner()` in order to remain consistent with the conventional naming of this functionality.

3.2 Pointless Proposal Creation

The `requestArbitration()` function creates a dxDAO "proposal" to call `disputeResolutionNotification()`, which is a `pure` function, and therefore has no visible side-effects by definition. It is not clear why the `requestArbitration()` function needs to create a proposal that does nothing as part of the arbitration process. (There is a good chance that this proposal could be elided entirely in favor of examination of the event logs emitted by the `requestArbitration()` function.)

3.3 Missing Check for Arbitrator Status

The `submitAnswerByArbitrator()` function does not check that the arbitrator for the given `questionId` is `address(this)`. Additionally, the function does not check that a fee was paid along with the question. It may make sense to check that the parameters to the `submitAnswerByArbitrator()` function are reasonable before passing them to `Realitio.submitAnswerByArbitrator()`.

3.4 Use `selector` for Function Selectors

The following code open-codes the calculation of a function selector:

```
bytes4(keccak256('disputeResolutionNotification(bytes32)'))
```

However, the following code is equivalent, and also robust to misspelling of the method name:

```
this.disputeResolutionNotification.selector
```

3.5 Inconsistent Solidity Version

The `DXdaoArbitrator.sol` contract uses solidity `0.8.0` or higher, but most recent by dxDAO uses the solidity `0.5.x` series compiler. New development ought to use the same compiler version(s) and flags so as to reduce cognitive overhead for developers when reviewing different pieces of code.

Also, note that solidity `0.8` uses checked math automatically, so using `SafeMath` (as the code does now) is unnecessary when using this compiler version.

3.6 Missing Documentation

The purpose of the `proposalDescriptionHash` and `metadata` fields of the `DXdaoArbitrator` contract is undocumented. Consider documenting these public contract variables.

It may also make sense to write documentation for the expected flow of calls from dxDAO and the `Avatar` contract to the `DXdaoArbitrator` contract, as it is not immediately obvious how these pieces are intended to fit together. The process of carefully documenting the intended interactions between those contracts may also reveal interoperability issues in the software architecture.

3.7 Superfluous Function

The `getDisputeFee()` function duplicates the auto-generated `disputeFee()` accessor function for the public `disputeFee` variable. The `getDisputeFee()` function can be removed entirely.