Secure Decentralized Solutions

Decentraland Security Audit

December 20, 2018 Mariana Soffer No Comments

Reading Time: 4 minutes

Contents

Introduction

CoinFabrik has been hired to audit the contracts for the Decentraland Land Auction. In the following sections we will provide a description of the contracts and their purpose, the audit methodology, detailed information about the issues found and, to wrap up, our conclusions on the contracts.

Overview

The contracts audited are from the "Land Auction" repository at https://github.com/decentraland/land-auction. The audit is based on the commit 76c575a27016fd643bcd3efd87438224744e1742, and updated to reflect changes at 05d3d4a7ec9c95ac5316e01ee4f6e417557b6ff7.

The contracts control the second Decentraland land auction. The land auction will begin the second week of December and it will last for 15 days. The auction will be implemented as a *Dutch auction*. It will provide a mechanism for both existing and new users to obtain unowned land. All remaining 9,300 unowned parcels will start at a price of 200,000 MANA. This price will then drop at a non-linear rate.

The audited contracts are:

- auction/LANDAuction.sol: Implementation of the auction contract
- auction/LANDAuctionStorage.sol: Data structures used by Auction contract
- dex/KyberConverter.sol: Converter to accept arbitrary tokens for payment
- dex/IKyberNetwork.sol: Interface for Kyber Network
- dex/ITokenConverter.sol: Interface for KyberConverter

Methodology

The following analyses were performed:

- Misuse of different call methods: call.value(), send() and transfer().
- Integer rounding errors, overflow, underflow and related usage of SafeMath functions.
- Old compiler version pragmas.
- Race conditions such as reentrancy attacks and front running.

- Misuse of block timestamps, assuming anything other than them being strictly increasing.
- Contract softlocking attacks (DoS).
- Potential gas cost of functions being over the gas limit.
- Missing function qualifiers and qualifier misuse.
- Fallback functions with a higher gas cost than allowed by a transfer or send call.
- Fraudulent or erroneous code.
- Code and contract interaction complexity.
- Wrong or missing error handling.
- Overuse of transfers in a single transaction instead of using withdrawal patterns.
- Insufficient analysis of function input requirements.

The contracts use Kyber Network to trade whitelisted ERC20 tokens for MANA tokens. In this audit we assume Kyber Network's contract behaves as defined by the *IKyberNetwork* interface.

Detailed findings

Critical severity

No issues with critical severity found.

Major severity

No issues with major severity found.

Minor severity

Minimum parcel price is not enforced

As per the Land auction documentation, the parcel will have a minimum price of 1,000 MANA. This minimum is not enforced in the *_setCurve* function in the *LANDAuction* contract.

```
function _setCurve(uint256[] _xPoints, uint256[] _yPoints) internal {
        uint256 pointsLength = _xPoints.length;
        require(pointsLength == _yPoints.length, "Points should have the same length");
        for (uint i = 0; i &lt; pointsLength - 1; i++) {
            uint256 x1 = _xPoints[i];
            uint256 x2 = _xPoints[i + 1];
            uint256 y1 = _yPoints[i];
            uint256 y2 = _yPoints[i + 1];
            require(x1 &lt; x2, "X points should increase");
            require(y1 &gt; y2, "Y points should decrease");
            (uint256 base, uint256 slope) = _getFunc(
                x1,
                x2,
                y1,
                y2
            );<br>
            curves.push(Func({
                base: base,
                slope: slope,
                limit: x2
            }));
        }
        initialPrice = _yPoints[0];
        endPrice = _yPoints[pointsLength - 1];
    }
```

Enhancements

Documentation is incomplete

Some interface contracts are not well documented, for example *IKyberNetwork* in the file *IKyberNetwork* and *LANDRegistry* in file *LANDAuctionStorage.sol*.

```
/**
    * @title Interface for contracts conforming to ERC-721
    */
    contract LANDRegistry {
        function assignMultipleParcels(int[] x, int[] y, address beneficiary) external;
    }
```

In contrast, others, such as *ITokenConverter*, have very good documentation.

Separate contracts into their own files

It would be more convenient for a couple of contracts in the file LANDAuctionStorage.sol to to have files of their own. Examples are *ERC20* and *LANDRegistry*

```
/**
    * @title ERC20 Interface with burn
    * @dev IERC20 imported in ItokenConverter.sol
    */
    contract ERC20 is IERC20 {
        function burn(uint256 _value) public;
    }
```

Non issues

Use of array item without checking length

In the *LANDAuction* contract constructor, an item in the array *_xPoints* is accessed without checking the length.

```
// Set total duration of the auction
        duration = _xPoints[_xPoints.length - 1];
```

This code is relying on current Solidity behavior: *_xPoints.length – 1* will cause an overflow and generate an invalid position, then the execution will fail when that position is dereferenced.

It is a non issue because in any case, the constructor will generate an error and the contract will fail to deploy, but it is better to have an explicit check for the array length.

Option to configure if tokens will be burned is unclear

In the function *allowToken*, the option *_shouldBurnTokens* appears to control whether the token will be burned or not.

In any case, the tokens will first be converted to MANA tokens using Kyber Network. When *_shouldBurnTokens* is set to true, approximately 5% of the tokens are reserved and will be burned directly. The other 95% are converted to MANA and burned in a similar manner.

When *_shouldBurnTokens* is set to false, the whole amount of tokens will be converted to MANA and burned in the *_processFunds* function.
The severity is minor since *allowToken* can only be called by the contract owner. In case of a mistake, the owner can call the function *disableToken* and reconfigure the token.

Rounding errors

The prices are set using a set of points, and intermediate values are calculated using linear interpolation. The *x* coordinate is the time in seconds since the UNIX epoch and the *y* coordinate is the price in MANA (which has a decimal precision of 18).

The slope is calculated as

Since it is stored in the contract as a *uint*, it will lose some of this precision. The UNIX timestamp on Friday November 30 at 12:00:00 UTC was 1543579200, which is about 1.5×10^9 seconds; also, MANA tokens use 18 decimal digits. The resulting slope will have a precision of around 10^9 decimal digits.

It should not cause problems because the minimum price was set to 1,000 MANA and the resulting rounding error should be around 10^{-12}, but we feel the need to mention it in case the parameters of the auction change.

Conclusion

The only issue we found was that the minimum value is not enforced when configuring the price "curve", but it should be pretty easy to fix.

Other findings listed in the *Non Issues* section do not cause direct harm because any unwanted behavior would require an explicit mistake made by the owner of the contracts.

We found the contracts to be very well developed and properly documented, with the exception of third party contracts such as the Kyber Network interface one.

Disclaimer: *This audit report is not a security warranty, investment advice, or an approval of the Decentraland Land Auction since Coinfabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.*

References

- Decentraland’s Second LAND Auction: https://decentraland.org/blog/announcements/announcing-decentralands-second-land-auction
- How will the LAND auction work?: https://decentraland.org/blog/technology/how-will-the-land-auction-work