DECENTER

Audit report for HashAuction

DECENTER

1. Summary

HashAuction Company has engaged Decenter in the period starting on September 17th and ending on September 19th 2018 to assess and audit their platform's Solidity smart contracts. This document describes the issues discovered during the audit. Decenter's assessment is focused primarily on code review of the contract, with an emphasis on security, gas usage optimization and overall code quality.

2. Audit

2.1 Authenticity

The audited contracts can be found in the Github repository: https://github.com/DecenterApps/HashAuction; the version used has commit hash b7634852dbd9c0f9f3cd977f7a7dc47af7c629c1.

2.2 Scope

This audit covered only the *.sol files linked to in the previous section.

2.3 Legend

🔴 High priority issues
🟠 Medium priority issues
🟡 Minor issues and optimisations
🟢 Notes and recommendations

3. Issues Found

1. Source of randomness used to mint hashes is predictable

File name: HashAuction.sol; (lines 122-126) generateRandomHash() function)

The purpose of this function is to return a random hash that cannot be predetermined and the sources currently used to generate this number are _seed, msg.sender, msg.data and current block number.

As of now the issue is that any user can precalculate hashess they would receive based on the block number.

In order to prevent predictability, we would suggest splitting the current createHash() function into two functions - createhash() and revealHash(), as one of the most secure solutions. The buyHash() function would only initiate the process and use the hash of the current block as the source of randomness, while revealHash() would simply mint items using a previously determined random seed.

2. Deposited funds could remain locked forever

File name: HashAuction.sol; (lines 154-170) revealVote() function

The only way for a voter to get back deposited tokens during the challenge period is if they call the revealVote() function.

The issue is that in case a voter doesn't reveal his vote during the reveal period, the funds they deposited during the vote commiting period will remain locked forever.

An addition of a simple withdraw function that would allow the voter to reclaim deposited funds after the reveal period is over would easily solve this issue.

3. Multiple deposit transfers can occur

File name: HashAuction.sol; (lines 172-180) depositForAuction() function

There is no guarantee that every voter will call the revealVote() function in the appropriate time during the challenge period, meaning some tokens can remain on the contract.

This issue would allow anyone to call the depositForAuction() function multiple times and drain these tokens from the contract.

Addition of an indicator that would show whether deposit for submission into TCR (Token Curated Registry) was made could be a solution for this.

4. Only the seller should be allowed to start an auction

File name: HashAuction.sol; (lines 189-204) startAuction() function

There is no check for who the person starting the auction is.

This means that anyone can call the startAction() function.

Adding a requirement for the person starting the auction to be a seller would solve this issue.

5. Owner has too much power over the contract

File name: HashAuction.sol; setAuctionCut() function

This setter is allowing the owner to set auction cut at any percentage.

By allowing this function in the contract, the owner would be able to manipulate the economy and discourage the authors of hashes to deposit for auction.

Adding additional check in the setter function to not allow setting a new auction cut if it’s greater than it previously was would encourage future hash authors to join the platform.

6. Possible race condition when buying from auction

Index changing is triggered every time an auction participant buys a hash at an auction.

Therefore auction participant could unintentionally purchase a hash that he didn’t want to.

This can be prevented by adding a third parameter in the buyAuction() function that would represent a hash address that an auction participant is trying to buy and comparing it against a hash that's at position _index in user's hashes array.

6. Same _secretHash can occur

File name: RegistryEntry.sol; (lines 149-158) commitVote() function

There is no check for uniqueness of the _secretHash parameter and it is currently possible for two exactly the same _secretHash parameters to occur if two users choose the same Secret.

Consequently, this means that in the event that a certain _secretHash was already used in the past, anyone can find the corresponding salt in the Ethereum transaction history and use it to determine their vote.

No two identical _secretHash parameters should be allowed and this can be ensured with inclusion of mapping that contains all previously used _secretHashes.

7. Reveal vote not possible for other addresses

File name: RegistryEntry.sol; (lines 160-177) revealVote() function

Adding the address parameter in input and replacing msg.sender with an address would allow anyone to reveal vote if they had the secret that corresponds with the address.

8. Older version of Solidity used

All .sol files.

An older version of Solidity is used. As of now the latest stable version is 0.4.25 and we recommend updating to and using the latest stable version.

4. Audit

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of business model, or any other statements about fitness of the contracts to purpose or their bugfree status. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

5. Closing Summary

It is the conclusion of this review that the codebase of the platform is well organized and sufficiently modular. In general, the smart contract code adopts all the relevant best practices and has very clean, legible and well-documented code. Aside for the remarks in this audit, the security of the contracts is sufficient given the assumed requirements. Main issues found by the audit are items marked as high priority.