G₀ Group

Security Review of

Conditional Tokens - Automated Market Makers

November 5, 2019

Overview

G0 Group was engaged to perform a security review of Automated Market Makers for Gnosis' Conditional Tokens Markets. G0 Group was contracted for a five person-week effort to that end. The primary subjects of this review were the smart contracts which implement an LMSR (logarithmic market scoring rule) based automated market maker for conditional token markets. This review was initially performed on https://github.com/gnosis/conditional-tokens-market-makers/tree/49b8fc1494a20f6233d10e534dac60ce7ccf9868.

Files in Scope

```
contracts/
    LMSRMarketMaker.sol
    LMSRMarketMakerFactory.sol
    MarketMaker.sol
    Whitelist.sol
```

Result Summary

During the course of this review, 2 issues were discovered and reported. One of these issues posed a direct security threat, the other concerned gas efficiency. All security issues reported have been remediated, and are not present in v1.4.0.

No further issues were discovered.

Issues

1. Re-entrancy issue allows theft of collateral

Type: *security* / ***Severity:*** *critical*

An attacker can steal collateral if they reenter `MarketMaker.trade()` after the last (to ensure maximum payout) execution of `MarketMaker.sol:line197` (using `MarketMaker.sol:line199`) before `mergePositionsThroughAllConditions()` is executed. Alternatively, an attacker can re-enter the contract during the first execution of `MarketMaker.sol:line199` before all of the position tokens have been transferred to the buyer. In both cases the contract will erroneously credit position tokens to the attacker's balance in the next (reentering) trade call. The re-entering trade can even be executed with all zero `outcomeTokenAmounts`.

Fix Description:

Issue was addressed in https://github.com/gnosis/conditional-tokens-market-makers/pull/22/ & https://github.com/gnosis/conditional-tokens-market-makers/pull/24 by adding reentrancy checks within the function and reordering the operations such that certain external calls which are at risk of reentracny (e.g. `collateralToken.transfer`) occur after the relevant contract state reads/writes.

2. Specific ordering of conditions during deployment of new market maker can lead to gas savings

Type: *note*

It's possible to minimise the amount of `splitPosition` calls if the `conditionIds` array provided to `LMSRMarketMakerFactory:createLMSRMarketMaker()` is ordered such that the ids of conditions with least outcomes are put before conditions with most outcomes.