SOLIDIFIED

Audit Report for Polymath. August 7, 2018.

Summary

Audit Report prepared by Solidified for Polymath covering the Polymath USDTieredSto module and it's auxiliary files.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the below token swap. The debrief took place on August 07, 2018 and the final results are presented here.

Audited Files

- USDTieredSTO.sol
- USDTieredSTOFactory.sol
- CappedSTO.sol
- CappedSTOFactory.sol
- MakerDAOOracle.sol
- PolyOracle.sol

Intended Behavior

The purpose of these contracts is to offer a tiered USD based Security Token Offering
The audit was based on commit.

Issues Found

Critical

No critical issues were found.

Major

No major issues were found.

Minor

1. Multiple issues in finalize function

a. `tempReturned` should keep track of tokens minted to the reserve address (it should be a sum of `remainingTokens`), instead it's just a sum of all tokens available for sale, as a result `finalAmountReturned` also ends up with an incorrect value

b. `tempSold` should keep track of amount of sold tokens (it should be a sum of `mintedPerTierTotal`), instead it keeps track of returned tokens, in result `finalAmountSold` also contains incorrect value

c. variable `finalAmountSold` only gets updated for tiers that weren't sold out, leading to this variable not reflecting the actual amount of tokens sold

d. Due to `mintedPerTierTotal` being set to the maximum value, function `capReached` will always return true after finalisation, even if the cap was not reached during the sale

e. It's unclear whether this function should only be called after the sale ends, but it can currently be called at any time. There's an unclear TODO comment in this function as well.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`..

2. The '_endTime' can be modified by pausing and unpausing

The implemented `modifyTimes()` function suggest that such alteration should only be done before the sale starts, but sale owners can postpone the end date at will by pausing the sale and the passing a new `endTime` when calling `unpause`.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

3. Owner can control oracle outcomes

In `MakerDAOOracle` the owner can set a manual fallback for the price and in `PolyOracle` he can set the `OracleURL` and the POLY-USD rate. While it is not a security issue by itself and those functions might be needed to manage the oracle, they offer a lot of power to control outcomes and in some situations that might be problematic. To mitigate, consider implementing some kind of delay when making significant changes, which will give users time to react.

Client's response:
We acknowledge that Polymath is able to arbitrarily (but transparently) modify the Oracle values. We note that we have split out permissions into admin (able to schedule new Oraclize calls) and owner (able to set manual prices) and that the owner address will be a multi-sig with internal controls around its security.

4. 'LatestScheduledUpdate' can end up having an incorrect values

This happens because the `_times` array in `schedulePriceUpdatesFixed` isn't required to be ascending order, making it possible to write incorrect values to `latestScheduledUpdate`.

Consider moving the `if statement` inside the `for loop` to mitigate this issue.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

5. Relative timestamps can break 'LatestScheduledUpdate' logic

If the `_times` arrays contains values that will be treated as relative timestamps by Oraclize (example: `_times = [50, 150, 250]`) it will break the logic of `LatestScheduledUpdate`, and the require statements, like `require(requestIds[_requestId] >= latestUpdate, "Result is stale");` and `require(requestIds[_requestId] <= now + oraclizeTimeTolerance, "Result is early");` in the `___callback()` function, specially if absolute timestamps haven't been used before.

Consider either fully supporting relative timestamps or banning it completely, by requiring `_times[i] >= now`.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

During the process of the audit, Polymath team found a new issue:

10 . Roundings can cause STO transaction to fail

The contract receive funds in either ETH or POLY, then it turn this into USD (to 18 decimal places of precision, rounding up or down as appropriate) and then turn this into a number of tokens that can be purchased (to 18 decimal places of precision, rounding up or down as appropriate).

To calculate whether we need to refund any ETH or POLY (i.e. if the full amount was not spent on purchasing tokens due to caps) we take the number of actually purchased tokens and calculate the corresponding USD value (to 18 decimal places of precision, rounding up or down as appropriate). Then this is converted back into ETH or POLY (and the difference with the sent funds is what is refunded)

Because of integer arithmetic in Solidity the rounding on the way from e.g. ETH > tokens meant that when converting back from tokens > ETH we could be off by 1 wei (or 1/10^18 POLY) which could cause an investment to revert.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

Notes

6. Sale tiers can be nonoptimal for buyers

If an STO is configured with undiscounted tokens in a given tier being more expensive than discounted tokens in the next tier, the sale can enter a state that is nonoptimal for buyers, in

which an investor purchases undiscounted tokens when there are still discounted tokens available in the next tier for a cheaper price. This may lead to investors postponing their purchase in an attempt to get a more favorable price.

Client's response:
We acknowledge this point - at the moment we think it is reasonable to allow issuers to set these values themselves, as they see fit. It is conceivable that an issuer may want to take this approach

7. Redundancy issues

a. In `_calculateTier`:

```
if (totalRemaining < discountRemaining)
   (tierSpentUSD, tierPurchasedTokens) = _purchaseTier(_beneficiary,
ratePerTierDiscountPoly[_tier], totalRemaining, _investedUSD, _tier);

else
   (tierSpentUSD, tierPurchasedTokens) = _purchaseTier(_beneficiary,
ratePerTierDiscountPoly[_tier], discountRemaining, _investedUSD, _tier);
spentUSD = spentUSD.add(tierSpentUSD);
```

Could be

```
if (totalRemaining < discountRemaining)
   (spentUSD, tierPurchasedTokens) = _purchaseTier(_beneficiary,
ratePerTierDiscountPoly[_tier], totalRemaining, _investedUSD, _tier);
else
   (spentUSD, tierPurchasedTokens) = _purchaseTier(_beneficiary,
ratePerTierDiscountPoly[_tier], discountRemaining, _investedUSD, _tier);
```

The add operation is uneccesary since `spentUSD` is always 0

b. in `getTokensSold`:
`getTokensSold` could call `getTokensMinted`

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

8. Give preference to modifiers

To make use of the `pausable` interface, most functions are checking the `paused` variable, but there's a `WhenNotPaused` modifier available.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

9. Remove unused code

`rmul()` and `rdiv()` functions are not used.

AMENDED [2018-8-29]:
This issue is no longer present in tag `v1.4.0`.

Closing Summary

Although no critical or major issues were found, there are some minor issues that can affect the desired behavior and it's recommended that they're addresses before proceeding to deployment.

AMENDED [2018-8-29]:
All of the issues raised in the report were fixed or addressed accordingly considering the trade-offs.

Audit Report for Polymath. August 7, 2018.

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Polymath platform or its products. This audit does not provide a security or correctness guarantee of the audited smart contracts. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

The individual audit reports are anonymized and combined during a debrief process, in order to provide an unbiased delivery and protect the auditors of Solidified platform from legal and financial liability.

Solidified Technologies Inc.