SOLIDIFIED

Audit Report for Polymath. October 18, 2018.

Summary

Audit Report prepared by Solidified for Polymath covering the Polymath Core 2.0.0 release.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the below token swap. The debrief took place on October 18, 2018 and the final results are presented here.

Audited Files

All the contracts present in Polymath Core 2.0.0 release.

Intended Behavior

The purpose of these contracts is to facilitate the creation and offering of regulatorily compliant security tokens.

The audit was based on commit `5fa2f1ea900075d615434f1adced2332a1878c9e`.

Issues Found

Major

Duplicates in excluded array can lead to overdrawing

Location

`DividenCheckpoint.sol`

Description
Having duplicates in the excluded array can lead to a artificial decrease in the calculated total supply, making possible for some investors to overdraw checkpoints rewards.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Percentage constraint can be broken

Description
The percentage constraint can be bypassed by burning tokens, which this module does not check for.

Location

```
PercentageTransferManager.sol
```

Recommendation
It's not clear whether this is a responsibility of this specific module, but having it installed can create the false sensation that token is protected.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Approve attack still possible

Location

```
SecurityToken.sol line 358
```

Description
The function acts as a wrapper around `increaseApproval` and `decreaseApproval`, which makes possible again for frontrunning approve attacks.

Recommendation
Split this function into separate functions `increaseModuleBudget` and `decreaseModuleBudget`

SOLIDIFIED

Audit Report for Polymath. October 18, 2018.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Allowed transaction might be rejected by transfers modules

Location

```
SecurityToken.sol line 534
```

Description

If all transfer managers return `NA`, the transaction won't be allowed. This means general transfer manager has to always be present. Right now, transfer managers that doesn't return `VALID` can't be used as standalone modules.

Recommendation

This could be fixed by getting rid of NA Result altogether, which should be possible.

Polymath's Response:

The GTM comes by default with the token as the whitelist is THE key feature of the ST.
IF for some reason the issuer removed the GTM and had let's say the CountTM, it's true all transactions will fail. This could be fixed by having a "ApproveAllTM" that basically returns true no matter what (which means everyone is on the whitelist).

Possible Sybil attack on tokens with CountTransferManager

Location

```
CountTransferManager.sol
```

Description

Trading can be blocked on tokens subject to `CountTransferManager` by a holder performing a sybil attack by redistributing his own tokens into multiple addresses.

Polymath's Response:

Sybil attacks are not technically possible with a permissioned token where each token holder must be KYC'ed before they can receive tokens. There is a possible issue if a single investor needs to KYC multiple accounts and split their balance across these accounts. This will be

addressed in a future release where we will track holding identities (which could map to multiple addresses) rather than holding addresses.

Minor

Signed messages can be replayed

Location

```
GenrealTransferManager.sol line 231
```

Description
Signed transactions can be replayed and as a consequence revert a change made by a `WHITELIST` manager

Recommendation
Consider adding a nonce in the signature process to avoid replays.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Issues with SingleTradeVolumeRestrictionManager.sol

Location

```
SingleTradeVolumeRestrictionManager.sol
```

Description
This module can be bypassed by wrapping the tokens in another erc20 compliant contract that uses batched transfer. Another option is to do multiple transactions.

Description
This module doesn't seem to be very effective, so maybe consider removing it or integrating its functionalities to a more powerful module.

Polymath's Response:
What is mentioned is true, to some extent since KYC restrictions would make it hard to do the wrapper approach. In any case, this TM is supposed to be combined with others that also limit the amount of tokens that can be transferred during a period of time for example.

We are leaving this module out of the release and moving it to an "experimental" folder for further review.

Issues with VolumeRestrictionTransferManager.sol

Location

```
VolumeRestrictionTransferManager.sol
```

Description
The implementation of the module can lead to unexpected results, like locking user tokens in the contract if the amount is less than the unlocked amount at a given time. Also, the expected usage of this module isn't very clear.

Polymath's Response:
The module has been moved to our "Experimental" folder and won't be deployed in its current state to mainnet.

Buyers should be able to set the maximum price they want to buy

Location

```
USDTieredSTO.sol
```

Description
Given the structure of the tiered STO, buyers can end up making purchases at a greater value than they initially desired

Recommendation
Consider implementing an optional parameter for a buyer to limit the maximum price for purchase

Polymath's Response:
This is a nice to have we'll be considering adding in a future upgrade of the module.

Archiving Permission Manager has no effect

Location

```
SecurityToken.sol
```

Description
If a user decides to archive the Permission Manager it won't have any practical difference because the modifier `withPerm` does not check weather the module is active.

Recommendation
Check if a permission manager is active before allowing or disallowing transactions

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Variable hashes should probably be precompiled into constants

Location
`SecurityTokenRegistry.sol`

Description
Computing the hash of storage keys at execution time is costly gas wise. Consider pre computing the hashes and saving it into constants.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Prunable investors can be missed

Location

```
SecurityToken.sol line 406
```

Description

Some prunable investor can be missed, if they are switched to an index previoulsy occupied by a pruned investor.

Polymath's Response:
We decided to remove the ability to prune `investors`. As a consequence this array becomes append-only and represents the list of all investors who ever held a non-zero balance.

We added `getInvestorsAt` to return a filtered list of investors who had a non-zero balance at the supplied checkpoint. This function is intended to be used off-chain as on-chain it could hit a block gas limit.

In order to be able to use a potentially large `investors` list on-chain, we added `iterateInvestors` function which takes a start and end index and iterates over `investors` between these indexes. This can always be used to iterate through `investors` as the gas is not a function of the length of the array. The current approach in DividendCheckpoint.sol which calls `getInvestors()` and then iterates over it locally would suffer as the entire array in copied in memory and hence the function is uncallable once the array hits a certain size. We added a disclaimer for this issue with the current implementation.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Modules needs more rigorous specifications

Description
It isn't clear how every module is supposed to work and how it should interact with other modules. Some of them only make sense if coupled with others and therefore is hard to determined if it is a intended behaviour or an unwanted side effect.

Polymath's Response:
It's true. We are starting to produce more comprehensive docs explaining how each module works.

“Read-only” fake transfers are confusing

Description
There's a variable that marks a transfer as non state modifying transaction, which is confusing. More clarification on the desired behavior of this would be helpful.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Polymath's Response:
In the future We should add more extensive documentation about TransferManagers to explain this with some examples, but on the code side I think this comment is sufficient.

Improvements and Optimizations

General

- Inconsistent use of kind vs safe math

AMEND[05-11-2018]
Issue has been fixed by Polymath team

- Inconsistent error messages (some describe what went wrong others describe the conditions it needs to pass)
- Off-chain storage seems more appropriate for metadata like delegateDetails/getInstructions/etc., however there's an understandable engineering trade-off here with complexity
- Boolean flags sometimes use modify pattern, sometimes add/remove pattern
- An analysis with regards to how many checkpoints it takes to make operations exceed the block gas limit should be done
- There are many unbounded loops that might hit the gas limit one day
- Consider packing variables of the same type together to save on gas, specially on structs
- Ordering of transfer managers has gas implications, so consider making it reorderable

ModuleRegistry.sol
L150, 182: Use the provided `owner()`

AMEND[05-11-2018]
Issue has been fixed by Polymath team

GeneralTransferManager.sol
L168: All STO sales are minting, so issuanceAddress will always be 0

Polymath's Response:
a) Currently the sales are minting but there can be a manual transfer STO.

b) Even if issuanceAddress is always 0x0, we need the check to make sure that the sender is actually issuanceAddress. At most, we can delete the issuanceAddress variable from storage and check the sender against hardcoded 0x0 address but I don't think we should be doing that.

SecurityToken.sol
L232 approve doesn't check if account has enough funds, so the require does nothing
L581: Verify transfer should probably be view
L635: onlyModuleOrOwner redundant
L672: onlyModule(BURN_KEY) doesn't make much sense, since this function is for users burning their own tokens and probably won't be called by modules
Modules could be stored in mappings instead of arrays to save gas

AMEND[05-11-2018]
Issues on lines 232, 635 and 672 have been fixed by Polymath team.

TrackedRedemption.sol
L48 This function seems to be unused

Polymath's Response:
`getPermission()` function is the mandatory function to make a module compliant with `IModule.sol` interface. So need to provide the body of that function whether it will be used or not.

ISTO.sol
L69: seemingly pointless function override

Polymath's Response:
We are not using the openzeppelin `Pausable` contract which inherits the Ownable contract by default. We have our own Pausable contract version which have internal function called `_unpause()` and `_pause()` which needs to be override to provide the onlyOwner permission.

PreSaleSTO
It can return wrong investorCount because it doesn't consider multiple investments from same person.

AMEND[05-11-2018]
Issue has been fixed by Polymath team

USDTieredSTO.sol

L279: why not mint all at once after the loop
L190-195: resetting the arrays seems unnecessary since before startTime they should be empty

AMEND[05-11-2018]
Issue L279 has been fixed by Polymath team

Polymath's Response regarding L190-195:
Initialising the array's with the length (`getNumberOfTiers()`) making the static array instead of dynamic.

GeneralPermissionManager.sol
L112: withPerm(CHANGE_PERMISSION) redundant
L163: why is securityToken from state not used?
L61: duplicates in allDelegates array can happen
Delegates can't be removed, but that's has no implications

AMEND[05-11-2018]
Issues have been fixed by Polymath team

ModuleFactory.sol
monthlySubscriptionCost and usageCost not implemented, also not enforcable right now

Polymath's Response:
`getPermission()` function is the mandatory function to make a module compliant with `IModule.sol` interface. So need to provide the body of that function whether it will be used or not.

TokenLib.sol
L95: function can be pure

Polymath's Response:
`getValueAt()` function reading the values from the environment (state values) so it can't be pure.

DividendCheckpoint.sol
L193: is this more gas efficient than just pushing?
Must be combined with transferral limitations to be effective.

Polymath's Response:
L193: It's not possible to do only one loop.

EtherDividendCheckpoint.sol
L108: could be optimized to only one loop

AMEND[05-11-2018]
Issue has been fixed by Polymath team

CappedSTO.sol
L154: Function undocumented and non-standard, present only in this STO
In a previous audit this module was

AMEND[05-11-2018]
Issue has been fixed by Polymath team

ManualApprovalTransferManager
Might be useful to index some event parameters

AMEND[05-11-2018]
Issue has been fixed by Polymath team

Disclaimer

Solidified audit is not a security warranty, investment advice, or an endorsement of the Polymath platform or Polymath's 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.

Audit Report for Polymath. October 18, 2018.

Solidified Technologies Inc.