G0 Group

Security Review of

Gnosis Dao Realitio Module

March 2021

Gnosis Dao Realitio Module / March 2021

Files in scope

All solidity files in:

https://github.com/gnosis/dao-module/tree/93166cd92793a7f7a0c1e5393981b072f55178c0

Current status

All found issues have been fixed.

Issues

1. When a proposal is being executed in executeProposalWithIndex, it's not validated that provided data matches originally submited proposal

Severity: critical

In `executeProposalWithIndex` the `txHashes` array which cotrols which transactions can be executed through the DAO is not checked against its hashed version in the `questionHashes` mapping set in `addProposalWithNonce`. This means that any accepted proposal can be used to execute arbitrary transactions through the DAO.

status - fixed

The issue is no longer present in

https://github.com/gnosis/dao-module/commit/7a5244d0a0a70b023d23af59659e0c055be7cca2

2. txHashes array can have duplicate members

Severity: minor

In `executeProposalWithIndex` and `addProposalWithNonce`, the `txHashes` array can contain duplicate hashes, which would allow some transactions to be executed in arbitrary order. This is not a vulnerability in the true sense, since it can be easily noticed when the hash array is validated on the client side.

status - fixed

The issue is no longer present in

https://github.com/gnosis/dao-module/commit/7a5244d0a0a70b023d23af59659e0c055be7cca2

3. DoS through forced out of gas exception in executeProposalWithIndex

Severity: major

In `executeProposalWithIndex` since return value of `executor.execTransactionFromModule` is not checked, attacker can provide just enough gas for the wallet transaction to fail, while the top level transaction succeeds preventing the accepted transaction from being successfuly executed.

status - fixed

The issue is no longer present in

https://github.com/gnosis/dao-module/commit/7a5244d0a0a70b023d23af59659e0c055be7cca2

Notes

4. passing txIndex to executeProposal will ensure constant gas cost

In `executeProposal` passing `txIndex` would allow to remove the `txHashes` search loop and achieve constant gas cost. Which ensures that every proposal can be executed inside the block limit no matter what size the `txHashes` array is.

5. oracle state variable should be replaced by an immutable

6. onlyExecutor modifier would make the code cleaner

There's a number of functions that check that `msg.sender == executor`, this check could be abstracted into a modifier making the code cleaner.

7. allowing execution of markProposalAsInvalid with only questionHash saves gas

Adding a version of `markProposalAsInvalid` that takes `questionHash` instead of `proposalId` and `txHashes` allows chepaer proposal invalidation.