Numer.ai Code Fix Review
v2.0

New Alchemy

April, 2018

Introduction

Numer.ai contacted New Alchemy about a proposed fix to an identified issue about the multiple ownership of their NMR smart contract[1] and its key rotation system.

The issue manifest itself in the fact that the `changeShareable` function from the `Shareable` contract does not clear the `owners` table prior to updating it and also does not clear the `ownerIndex` mapping which is the only requirement needed to validate the `onlyOwner` modifier (which calls the `isOwner` function to check that `ownerIndex[msg.sender] > 0`).

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

Proposed Changes

Numer.ai proposed the following change to the `NumeraireDelegate` contract which can be updated through the `NumeraireBackend changeDelegate` function.

```
118 function numeraiTransfer(address _to, uint256 _value) onlyManyOwners(sha3(msg.data)) returns (bool c
119     if (_value == 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF) {
120         ownerIndex[_to] = 0;
121         return true;
122     }
123
124     // Check for sufficient funds.
125     require(balanceOf[numerai] >= _value);
126
127     balanceOf[numerai] = safeSubtract(balanceOf[numerai], _value);
128     balanceOf[_to] = safeAdd(balanceOf[_to], _value);
129
130     // Notify anyone listening.
131     Transfer(numerai, _to, _value);
132
133     return true;
134 }
```

Numera.ai also proposed to lock down the `createTournament` and `createRound` functions (`contract/contracts/NumeraireDelegate.sol`) by adding the `onlyOwner` modifier to these two functions prototypes.

[1] https://github.com/numerai/contract

Review Results

The proposed change to `numeraiTransfer` while hijacking the original intended purpose of the function would effectively allow clearing specific entries of the `ownerIndex` mapping. If called on the addresses to be removed from the owner list, this effectively fixes the issue of allowing these addresses to continue calling functions using the `onlyOwner` modifier or the `onlyManyOwners` modifier.

However, New Alchemy strongly advises against implementing the changes as proposed for the following reason. This change creates potential conditions for unrecoverable errors: calling the `numeraiTransfer` function to revoke an owner out of order (before changing the owners through `changeShareable`) or calling it by mistake on one of the current legitimate owner will lower the number of owners. If the number of owners falls below the stored `required` value, then all functions relying on the `onlyManyOwners` modifier will become impossible to call, including functions to update the `owners` list and `required` number and the `numeraiTransfer` function. The risks of creating unrecoverable locking conditions through user mistakes are real.

One potential way of ensuring such locking conditions are prevented would be to check that the ownerIndex being set back to zero is not part of the first `required` number of owners in the `owners` list (which should correspond to the legitimate owners). this could be implemented in the following manner:

```
118 function numeraiTransfer(address _to, uint256 _value) onlyManyOwners(sha3(msg.data)) returns (bool c
119     if (_value == 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF) {
120         if((address(owners[ownerIndex[_to]]) != _to) || (ownerIndex[_to] > (required + 1)) {
121             ownerIndex[_to] = 0;
122         }
123         return true;
124     }
125
126     // Check for sufficient funds.
127     require(balanceOf[numerai] >= _value);
128
129     balanceOf[numerai] = safeSubtract(balanceOf[numerai], _value);
130     balanceOf[_to] = safeAdd(balanceOf[_to], _value);
131
132     // Notify anyone listening.
133     Transfer(numerai, _to, _value);
134
135     return true;
136 }
```

Please note that the above code would still allow removing legitimate owners if the `required` number of owners is lower than the amount of legitimate owners in the list. It will however prevent having less owners than the `required` number to validate a `onlyManyOwners` modified function.

The changes to add the `onlyOwner` modifier to the `createTournament` and `createRound` functions will effectively lock down these two functions and prevent calls to each of these from successfully terminating if called by non-owners of the contracts.

Second Review Notes: The above code had a missing closing parenthesis in the added `if` statement as was accurately noticed by the Numearai team. Numerai also commented that the second condition of the `if` statement (`(ownerIndex[_to] > (required + 1))`) contains an off-by-one error that prevents removing the first unwanted index from the list. The conditional statement with these added corrections should read as follows:

```
if((address(owners[ownerIndex[_to]]) != _to) || (ownerIndex[_to] > required)) {
    ownerIndex[_to] = 0;
}
```

Furthermore, Numerai proposed to modify the `if` statement:

```
if (address(owners[ownerIndex[_to]]) != _to) {
```

Numerai gave the following argument for the change: This would allow reducing the number of owners (and required owners) as the previous condition would have prevented such a change because the index and corresponding address would still be valid if the owners list is not overwritten with additional zeros where the previous owners address were stored. Numerai is absolutely correct in this instance, and this change would have the intended effect of allowing reduction of the number of owners while successfully preventing previous owners from calling `onlyOwner` and giving their authorization for `onlyManyOwners` calls.

To allow this change to be used in an effective manner, Numerai indicated that calls to `changeShareable` would have to be made in a manner to effectively overwrite previous unwanted addresses in the owners list, such as : `changeShareable(_owners=[1b, 0, 0, 0, 0], _required=1)`. This call will effectively remove previous extra addresses from the owners list with the following caveats :

- the caller must ensure that the new owner list contains enough trailing zeros to effectively remove all the previous extra owners from the list.
- the caller must ensure that the `_required` parameter is absolutely accurate and represent a number equal or lower than the number of non-zero owner addresses (and unique addresses) from the new `_owners` list. Failure to do so (e.g.: `changeShareable(_owners=[1b, 0, 0, 0, 0], _required=2)`) would still allow the call to succeed but would effectively lock the contract and prevent any further call to any `onlyManyOwners` functions as the number of required owners would now be greater than the number of unique addresses in the `owner` list.

While the above condition represents a real risk of contract lockout for Numerai, the risk already exists in the current deployment. New Alchemy believes that the proposed changes from Numerai will not add any extra risk to these contracts but will actually improve the safety of these by allowing the removal of previous unwanted owners and allowing reducing the number of required owners for `onlyManyOwners` functions while keeping an accurate `owner` list and `ownerIndex` list.

