SOLIDIFIED

Audit Report for FOAM. November 26, 2018.

Summary

Audit Report prepared by Solidified for FOAM covering the first iteration of their signaling protocol.

Process and Delivery

Three (3) independent Solidified experts performed an unbiased and isolated audit of the contracts below. The debrief took place on November 23, 2018 and the final results are presented here.

Audited Files

The following files were covered during the audit:

```
contracts
 ├──── CSTRegistry.sol
 ├──── ERC721Controllable.sol
 ├──── SignalToken.sol
 ├──── StakeToken.sol
 └──── FoamTokenController.sol
```

Intended Behavior

The purpose of this first generation of signaling contracts is:
- To bond the FOAM token to indicate where miners are to be located
- To record signaling for future rewards of early signalers
- Perform proof-of-use.
A full specification is available here.

The audit was based on commit `ce125f6525efa2602b841ac91b16a3e975609db2`.

Issues Found

Notes, Improvements, & Optimizations

1. Wrong Commenting and Parameter Naming in `transferFrom` of `ERC721Controllable`

Description

The overridden `transferFrom` function in `ERC721Controllable` is defined as follows:

```
/// @dev It calls parent StandardToken.transferFrom() function. It will ///
transfer from an address a certain amount of tokens to another address
/// @return True if the token is transfered with success
    function transferFrom(address _from, address _to, uint256 _value)
        public
        isAllowed(_from, _to)
    {
        super.transferFrom(_from, _to, _value);
    }
```

The commenting and the naming of last parameter (`_value`) suggest ERC-20 behaviour, whereas in an ERC-721 token the last parameter should be the token id.

Recommendation

The above code works, but the comments should be changed and the parameter renamed to avoid confusion.

2. Violation of Checks-Effects-Interactions Pattern

Description

The following function of the `StakeToken` contract performs state changes after an external call:

```
function mint(address owner, uint256 stake) public returns (uint256
tokenID) {
    require(msg.sender == owner, "msg.sender == owner");
    require(intrinsicToken.transferFrom(owner, this, stake),
"transferFrom");
    nftNonce += 1;
    tokenID = nftNonce;
    tokenStake[tokenID] = stake;
    tokenMintedOn[tokenID] = block.timestamp;
    super._mint(owner, tokenID);
    return tokenID;
  }
```

Recommendation
To remedy this, the call to `transferFrom` should be moved to the end of the function. While the code being called is trusted, and should not cause any issues; it is recommended to avoid state changes after external calls.

3. Use of `owner` in `StakeToken` potentially confusing

Description
`StakeToken` inherits an `owner` storage variable. Therefore, using `owner` as a parameter/function scoped variable in functions `mint` and `burn` could lead to miscommunication.

Recommendation
The compiler resolves these instances as intended, however renaming them to distinguish them from the "owner" of the contract is recommended.

Closing Summary

No immediate security issues were discovered in the course of the audit. The ERC20 and ERC721 implementations have been verified as fully compliant with their respective standards.

Disclaimer