RegisterNameCrossChainExecutor Audit Report

Date: 2025-12-14

Auditor: XinghuiChen

Email: 0xBitcoiner@protonmail.com

Summary

Scope

https://github.com/decentraland/offchain-marketplace-contract/blob/1b0319e53d0554ba987c8bbdb4621ad434476abe/src/credits/executors/RegisterNameCrossChainExecutor.sol

The contract under audit is `RegisterNameCrossChainExecutor.sol`.

Findings

Critical	High	Medium	Low	Informational	Optimizations	Recommendations
-	-	-	-	3	1	3

Informational

1. Line 78, `in 8 decimals` should be `in 18 decimals`.
2. Line 119, `external payable` should be `public payable`.
3. Line 126, `_args.expiresAt < block.timestamp` can be changed to `block.timestamp >= _args.expiresAt` (i.e., revert when `timestamp == expiresAt`), to keep the logic consistent with other contracts.

Optimization

1. Under the current logic, for the `ExternalCall` struct used in the `execute` function:
   - The `target` must be `coral`, since `coral` is immutable, the `target` field can be removed.
   - The `selector` must be `0x58181a80`, so the `selector` field can also be removed, and it can be merged into `data`.
   - `manaFee` can be calculated from `data`, so the `extra` field can also be removed.
   - Can decode `data` and then verify each field to ensure security.

   Then, since this `ExternalCall` struct has only a few fields, it can be removed.

   Changes:

   - Add `import {ISquidMulticall} from "src/credits/executors/ISquidMulticall.sol";` (ISquidMulticall.sol)
   - Remove the `ExternalCall` struct.
   - Remove the parameters of `event Executed()`, `error ExecutionExpired()`, and `error CallFailed()`.
   - Add `error InvalidCallData(uint256 _flag);`
   - Update the `execute` function to the following:

```
function execute(uint256 expiresAt, bytes calldata callData) external
nonReentrant whenNotPaused {
    if (_msgSender() != creditsManager) {
        revert Unauthorized(_msgSender());
    }
    if (block.timestamp >= expiresAt) {
        revert ExecutionExpired();
    }

    if (bytes4(callData) != 0x58181a80) {
        revert InvalidSelector();
    }
    (address decMana, uint256 decPrice, ISquidMulticall.Call[] memory decCalls) =
abi.decode(callData[4:], (address, uint256, ISquidMulticall.Call[]));
    if (decMana != address(mana)) {
```

```
            revert InvalidCallData(1);
        }
        if (decCalls.length != 1) {
            revert InvalidCallData(2);
        }
        if (decPrice <= NAME_PRICE) {
            revert InvalidCallData(3);
        }
        if (decCalls[0].target != NameRegistry_Address) {
            revert InvalidCallData(4);
        }
        if (bytes4(decCalls[0].callData) != NameRegistry_RegisterName_Selector) {
            revert InvalidCallData(5);
        }

        _validateMANAFee(decPrice - NAME_PRICE);

        mana.forceApprove(coral, decPrice);
        mana.transferFrom(creditsManager, address(this), NAME_PRICE);
        (bool success,) = coral.call(callData);
        if (!success) {
            revert CallFailed();
        }
        mana.forceApprove(coral, 0);
        emit Executed();
    }
```

In addition, since the `ExternalCall` struct in `CreditsManagerPolygon.sol` already includes a check for `expiresAt`, the `expiresAt` parameter here can be considered for removal.

Recommendations

1. The `execute` function doesn't validate `_args.data`. Since the signature must be created by someone who has the `EXTERNAL_CALL_SIGNER_ROLE` permission, this is not a major issue. However, validating the `data` parameter would be safer.
2. In `_validateMANAFee` function, It would be better to check `_manaFee > 0` .
3. Line 137, `transferFrom` can be changed to `safeTransferFrom`, to keep the logic consistent with other contracts.