Credits Manager Audit Report

Date: 2025-05-16
Auditor: XinghuiChen
Email: 0xBitcoiner@protonmail.com

Summary

Scope

https://github.com/decentraland/offchain-marketplace-contract/commit/4fb7e3db9939fc226c22b6568338926570f249f9

The contract to be audited is `CreditsManagerPolygon.sol` .

Findings

Critical	High	Medium	Low	Informational	Optimizations	Recommendations
-	-	-	1	4	5	8

Low

1. In `_handleMarketplacePreExecution`, there is no check to ensure `asset.assetType` is ether `ASSET_TYPE_ERC721` or `ASSET_TYPE_COLLECTION_ITEM`. In extreme cases, users can set it with `ASSET_TYPE_ERC20` or `ASSET_TYPE_USD_PEGGED_MANA`, which leads to:

   - Transaction gets passed even `primarySalesAllowed` and `secondarySalesAllowed` are false.
   - Fee gets transferred to `DecentralandMarketplacePolygon's feeCollector` instead of `royaltyBeneficiary`.

   POC is in the attachment(`poc1.txt`). So it's better to fix as below:

   ```
   if (asset.assetType == ASSET_TYPE_ERC721) {
       if (!memSecondarySalesAllowed) {
           revert SecondarySalesNotAllowed();
       }
   } else if (asset.assetType == ASSET_TYPE_COLLECTION_ITEM) {
       if (!memPrimarySalesAllowed) {
           revert PrimarySalesNotAllowed();
       }
   } else {
       revert InvalidTrade(trade);
   }
   ```

   Result: Fixed.

Informational

1. In the comment of `revokeCustomExternalCallSignatures` funtion, `Only the owner and credits revoker..` should be `Only the owner and external call revoker..`.

   Result: Not change.

2. In the comment of `revokeCreditSignatures` function, `Revokes a credit` should be `Revoke credits`.

   Result: Changed.

3. In `_handleCollectionStorePreExecution`, it's safer to check each item of `itemToBuy.beneficiaries` not be address(0). If indeed not needed, the `itemToBuy` variable can be deleted as it's used only once.

   Result: Not change.

4. Please be aware that: if `CREDITS_SIGNER_ROLE` gets changed(e.g., admin calles `revokeRole` then `grantRole` to new address), all the previously signed credit signatures(include partially spent credits) are invalid.

   Result: ACK.

Optimization

1. In `_handleMarketplacePreExecution` function, the param can be changed to `calldata`.

   Result: Not change.

2. In `_validateAndApplyCredits`, can cache `_args.credits.length` instead of reading multiple times(3 places).

   Result: Not change.

3. The `InvalidCreditValue` error can be removed logically, as `credit.value == 0` will trigger `CreditConsumed` too.

   Result: Not change.

4. In `_executeExternalCall` and `_handleCustomExternalCallPreExecution`, as `_args.externalCall` is used multiple times, it's better to define a variable first. i.e.,

   ```
   ExternalCall calldata externalCall = _args.externalCall;
   // Use externalCall instead of _args.externalCall
   ```

   Result: Not change.

5. The `useCredits` function has a limitation: If `currentHourCreditableManaAmount` is $90, user wants to use $100 credits to buy a $100 item, even he sets `maxUncreditedValue` with $10, the transaction will fail(`MaxManaCreditedPerHourExceeded`). As the principle is to prioritize deducting as much credits as possible from `_args.credits`, the `maxCreditedValue` field of struct `UseCreditsArgs` can be removed, this also removes the limitation. i.e.,

   - Remove the `uint256 maxCreditedValue` field in `UseCreditsArgs` struct.
   - Remove the `_validateAndApplyCredits` function.
   - Remove the `MaxCreditedValueZero` error.
   - Remove the `MaxCreditedValueExceeded` error.
   - Remove the `MaxManaCreditedPerHourExceeded` error.
   - Change function as below:

     ```
     function useCredits(UseCreditsArgs calldata _args) external nonReentrant
     whenNotPaused {
         address sender = _msgSender();
         // Safer to validate argument first
         (uint256 totalRemainingCreditValue, bytes32[] memory creditSignatureHashes) =
     _validateCredits(_args, sender);

         _handlePreExecution(_args, sender);

         uint256 currentHourCreditableManaAmount =
     _computeCurrentHourCreditableManaAmount();
         // Need verify currentHourCreditableManaAmount > 0?
         uint256 maxUsableCredits = totalRemainingCreditValue <=
     currentHourCreditableManaAmount ? totalRemainingCreditValue :
     currentHourCreditableManaAmount;
     ```

```
    uint256 manaTransferred = _executeExternalCall(_args, sender,
maxUsableCredits);

    uint256 creditedValue = _applyCredits(_args, sender, manaTransferred,
maxUsableCredits, totalRemainingCreditValue, creditSignatureHashes);
    _handleUncreditedValue(_args, manaTransferred - creditedValue, sender);

    _handlePostExecution(_args, sender);
}

function _validateCredits(UseCreditsArgs calldata _args, address _sender)
internal view returns (uint256 totalRemainingCreditValue, bytes32[] memory
creditSignatureHashes) {
    uint256 creditsLength = _args.credits.length;
    if (creditsLength == 0) {
        revert NoCredits();
    }
    if (creditsLength != _args.creditsSignatures.length) {
        revert InvalidCreditsSignaturesLength();
    }
    creditSignatureHashes = new bytes32[](creditsLength);
    for (uint256 i = 0; i < creditsLength; ++i) {
        Credit calldata credit = _args.credits[i];
        bytes32 signatureHash = keccak256(_args.creditsSignatures[i]);
        if (block.timestamp > credit.expiresAt) {
            revert CreditExpired(signatureHash);
        }
        if (isRevoked[signatureHash]) {
            revert RevokedCredit(signatureHash);
        }
        address recoveredSigner = keccak256(abi.encode(_sender, block.chainid,
address(this), credit)).recover(_args.creditsSignatures[i]);
        if (!hasRole(CREDITS_SIGNER_ROLE, recoveredSigner)) {
            revert InvalidSignature(signatureHash, recoveredSigner);
        }
        uint256 remainingValue = credit.value - spentValue[signatureHash];
        if (remainingValue == 0) {
            revert CreditConsumed(signatureHash);
        }
        totalRemainingCreditValue += remainingValue;
        creditSignatureHashes[i] = signatureHash;
    }
}

function _applyCredits(UseCreditsArgs calldata _args, address _sender, uint256
_manaTransferred, uint256 _maxUsableCredits, uint256 _totalRemainingCreditValue,
bytes32[] memory _creditSignatureHashes) internal returns (uint256 creditedValue)
{
    if (_maxUsableCredits <= _manaTransferred && _maxUsableCredits ==
_totalRemainingCreditValue) { // Credits fully used(Most time)
        creditedValue = _maxUsableCredits;
        for (uint256 i = 0; i < _creditSignatureHashes.length; ++i) {
            Credit calldata credit = _args.credits[i];
            bytes32 signatureHash = _creditSignatureHashes[i];
            emit CreditUsed(_sender, signatureHash, credit, credit.value -
spentValue[signatureHash]);
            spentValue[signatureHash] = credit.value;
        }
    } else { // Credits partially used
        // 3 situations:
        // _manaTransferred=100, _maxUsableCredits=90,
_totalRemainingCreditValue=100
        // _manaTransferred=50, _maxUsableCredits=90,
_totalRemainingCreditValue=100
        // _manaTransferred=50, _maxUsableCredits=90,
_totalRemainingCreditValue=90
        creditedValue = (_maxUsableCredits <= _manaTransferred ?
_maxUsableCredits : _manaTransferred);
        uint256 totalRemainingValue = creditedValue;
        for (uint256 i = 0; i < _creditSignatureHashes.length; ++i) {
            Credit calldata credit = _args.credits[i];
            bytes32 signatureHash = _creditSignatureHashes[i];
            uint256 creditRemainingValue = credit.value -
spentValue[signatureHash];
            if (creditRemainingValue < totalRemainingValue) {
                spentValue[signatureHash] = credit.value;
                totalRemainingValue -= creditRemainingValue;
                emit CreditUsed(_sender, signatureHash, credit,
creditRemainingValue);
            } else {
                spentValue[signatureHash] += totalRemainingValue;
                emit CreditUsed(_sender, signatureHash, credit,
totalRemainingValue);
                break;
            }
        }
    }
    manaCreditedThisHour += creditedValue;
    emit CreditsUsed(_sender, _manaTransferred, creditedValue);
}

// replace _currentHourCreditableManaAmount param with maxUsableCredits, need
change comment
function _executeExternalCall(UseCreditsArgs calldata _args, address _sender,
uint256 maxUsableCredits) .. {
    if (_args.maxUncreditedValue > 0) {
        mana.safeTransferFrom(_sender, address(this), _args.maxUncreditedValue);
    }
    mana.forceApprove(_args.externalCall.target, _args.maxUncreditedValue +
maxUsableCredits);
    ..
}
```

Result: Not change.

Recommendations

1. In `ILegacyMarketplace.sol`, `safeExecuteOrder` can be removed as it's not used.

   Result: Not change.

2. In `denyUsers` function, an option is to check the length of the two parameters are equal.

   Result: Changed.

3. An option is to use `ERC721Holder` instead of `IERC721Receiver`, and remove the `onERC721Received` function in `CreditsManagerPolygon.sol`.

   Result: Not change.

4. `CustomExternalCallNotAllowed` defination can be removed and use `revert InvalidExternalCallSelector(..)` instead.

   Result: Not change.

5. In the comment of `_handlePreExecution` function, it's better to change to `The caller of the useCredits function` to make it consistent with other places.

   Result: Not change.

6. `EXTERNAL_CALL_SIGNER_ROLE` can be renamed to `CUSTOM_EXTERNAL_CALL_SIGNER_ROLE` as it's only used for custom external call. Same for `EXTERNAL_CALL_REVOKER_ROLE`.

   Result: Not change.

7. For the error `ExternalCallFailed`, add an error reason param may help debug. i.e.,

   - Change to `error ExternalCallFailed(ExternalCall _externalCall, bytes retData);`
   - In `_executeExternalCall`, change to

     ```
     (bool success, bytes memory retData) =
     _args.externalCall.target.call(abi.encodePacked(_args.externalCall.selector,
     _args.externalCall.data));
     if (!success) {
         revert ExternalCallFailed(_args.externalCall, retData);
     }
     ```

   Result: Not change.

8. The `CreditUsed` event already logs `_creditId`, worth considering whether it's needed to log the whole `Credit` struct too.

   Result: Not change.