Offchain-marketplace Audit Report

Date: 2024-09-07

Auditor: XinghuiChen

Email: 0xBitcoiner@protonmail.com

Summary

Scope

https://github.com/decentraland/offchain-marketplace-contract/tree/f9f658112c02570101c1a2c791215f5184af4cf1

Smart contracts found in the `src/` directory, excluding `src/mocks/`.

Findings

Critical	High	Medium	Low	Informational	Optimizations	Recommendations
1	-	-	-	3	23	2

Critical

1. In `_updateAssetWithConvertedMANAPrice` of `AggregatorHelper.sol`, should be:

   ```
   _asset.value = _asset.value * 1e18 / uint256(_manaUsdRate);
   ```

   Since MANA/USD means 1 MANA equals x USD, if want to sell for y USD, it's y/x MANA.

   Result: Fixed.

Informational

1. As `DecentralandMarketplacePolygon` supports meta-transaction, if the contractAddress of `ExternalCheck` is the contract itself, `msgSender()` function will return an error address. For example:

   - In `DecentralandMarketplacePolygon.t.sol`, import `Marketplace` file and append next function:

     ```
     function test__getMsgSenderNotRight() public {
         Marketplace.Trade[] memory trades = new Marketplace.Trade[](1);
         trades[0].signer = signer.addr;
         trades[0].checks.expiration = block.timestamp;
         trades[0].checks.externalChecks = new Marketplace.ExternalCheck[](1);
         trades[0].checks.externalChecks[0].contractAddress = address(marketplace);
         trades[0].checks.externalChecks[0].selector = marketplace.pocFunc.selector;
         trades[0].checks.externalChecks[0].value = type(uint256).max;
         trades[0].checks.externalChecks[0].required = true;
         trades[0].signature = signTrade(trades[0]);
         vm.prank(other);
         marketplace.accept(trades);
     }
     ```

   - In `DecentralandMarketplacePolygon.sol`, import `console` and add next function:

     ```
     function pocFunc(address _signer, uint256 value) external view returns (bool) {
         console.log('msg.sender:', msg.sender);
         console.log('_getMsgSender():', _getMsgSender());
         return true;
     }
     ```

   Run test:

   ```
   [PASS] test__getMsgSenderNotRight() (gas: 89417)
   Logs:
     msg.sender: 0xF62849F9A0B5Bf2913b396098F7c7019b51A820a
     _getMsgSender(): 0xFFfFfFffFFfffFFfFFfFFFFFffFFFffffFfFFFfF //
   uint160(trades[0].checks.externalChecks[0].value)
   ```

It's better to make sure `contractAddress` must not be the marketplace contract itself, but currently there isn't much need.

Result: Not need change. `If there is no issue here. Adding an extra check would only increase gas.`

2. On Polygon, seller may get less when sell different type NFTs in a single trade:

```
// Bulk sell
trade.sent[0] = "NFT1"  // NFT1 has no royaltyBeneficiary
trade.sent[1] = "NFT2"  // NFT2 has royaltyBeneficiary
trade.received[0] = "200 MANA"
// Seller get: 200 - 200 * 2.5%(feeCollector) - 200 * 2.5%(royaltyFees) = 190 MANA

// Single sell NFT1
trade1.sent[0] = "NFT1" // NFT1 has no royaltyBeneficiary
trade1.received[0] = "100 MANA"
// Single sell NFT2
trade2.sent[0] = "NFT2" // NFT2 has royaltyBeneficiary
trade2.received[0] = "100 MANA"
// Seller get: 100 - 100*2.5(feeCollector) + 100 - 100*2.5(royaltyFees) = 195 MANA
```

Same effect for different asset types(e.g., `sent[0]` is `ASSET_TYPE_ERC721`(has royaltyBeneficiary) and `sent[1]` is `ASSET_TYPE_COLLECTION_ITEM`).

Result: Not need change(as a feature).

3. Please be aware that:
    - In `CouponManager.sol`, coupon's `signatureUses` will increase even `trade.accept` is empty.

      Result: Fixed.
    - `TradeId` aren't same even only the order of `Received Assets` not same.

      Result: Not need change. `The Trade Id is purely something for UX on our dApps so it is expected to be handled correctly by us.`

Optimization

Global

1. For the verifying trade process, params can all be changed from `memory` to `calldata` to save gas:

File	Lines
CommonTypesHashing.sol	14, 18, 34
Signatures.sol	64
Verifications.sol	32, 73, 92, 97
CouponManager.sol	65, 111
Marketplace.sol	39, 65, 69, 93, 114
MarketplaceTypesHashing.sol	19, 23, 39, 43, 60
CouponTypesHashing.sol	12

Besides, some test files need change to `calldata` too:

- DecentralandMarketplaceEthereum.t.sol, line 45
- DecentralandMarketplacePolygon.t.sol, line 47, 68
- Marketplace.t.sol, line 24
- CouponManager.t.sol, line 14

Result: Changed.

2. `MerkleProof.verify` can be simplified:

```
// In Verifications.sol
if (!MerkleProof.verify(_allowedProof, _allowedRoot,
keccak256(abi.encodePacked(_caller))))

// In CollectionDiscountCoupon.sol
if (!MerkleProof.verify(callerData.proofs[i], data.root,
keccak256(abi.encodePacked(collectionAddress))))
```

For more detail, please see: https://medium.com/block6/using-merkle-trees-in-solidity-64409513989a

Result: Not need change. `https://github.com/OpenZeppelin/merkle-tree?tab=readme-ov-file#validating-a-proof-in-solidity double hash to prevent second pre image attacks`.

3. `Overriden` should be `Overridden`(4 occurs).

   Result: Fixed.

4. It's better to lock pragma(just like in `NativeMetaTransaction.sol`).

   Result: Changed.

NativeMetaTransaction.sol

1. The `_verify` function's logic can be moved into `executeMetaTransaction`, and remove the defination of `MetaTransaction` :

```
function executeMetaTransaction(..) {
        bytes32 structHash = keccak256(abi.encode(META_TRANSACTION_TYPEHASH,
nonces[_userAddress], _userAddress, keccak256(_functionData)));
        if (_userAddress != ECDSA.recover(_hashTypedDataV4(structHash), _signature))
{
            revert MetaTransactionSignatureDoNotMatch(); // Add a new error
        }
    ++nonces[_userAddress];
    ..
```

Result: Changed.

2. The last return line of `_getMsgSender` can be removed.

   Result: Changed.

AggregatorHelper.sol

1. In `_updateAssetWithConvertedMANAPrice`, as `_asset` is passed by reference, the return type and return line can be removed. The calls in `DecentralandMarketplacePolygon` and `DecentralandMarketplaceEthereum` need get changed correspondly.

   Result: Changed.

Marketplace.sol

1. The `usedTradeIds` is mainly for auction(diffrent trades, same caller); For common listing-buy process, it seems useless(`checks.use` is enough). Maybe it's better to only deal it in specific condition. For example:

```
// Only deal tradeId when checks.uses == type(uint256).max
function _verifyTrade(.._trade, .._caller) internal {
    bytes32 hashedSignature = keccak256(_trade.signature);
    address signer = _trade.signer;
    _verifyChecks(_trade.checks, hashedSignature, signatureUses[hashedSignature],
signer, _caller);
    _verifyTradeSignature(_trade, signer);
    if (_trade.checks.uses == type(uint256).max) {
        bytes32 tradeId = getTradeId(_trade, _caller);
        if (usedTradeIds[tradeId]) {
            revert UsedTradeId();
        }
        usedTradeIds[tradeId] = true;
    }
    ++signatureUses[hashedSignature];
}
```

Result: ACK.

2. The `cancelSignature` function can be much cheaper if the type of `cancelledSignatures` is:

   ```
   mapping(address => mapping(bytes32 => bool)) public cancelledSignatures;
   ```

   But meanwhile the `accept` function will cost a little more gas. Detail:

   - In `Signatures.sol`, change to `mapping(address => mapping(bytes32 => bool)) public cancelledSignatures;`
   - In `Signatures.sol`, change function `_cancelSignature` as below:

     ```
     function _cancelSignature(address _caller, bytes32 _hashedSignature) internal {
         cancelledSignatures[_caller][_hashedSignature] = true;
         emit SignatureCancelled(_caller, _hashedSignature);
     }
     ```

   - In `Verifications.sol`, change to `if (cancelledSignatures[_signer][_hashedSignature]) {`
   - In `CouponManager.sol` and `Marketplace.sol`, change function `cancelSignature` as below:

     ```
     function cancelSignature(bytes32[] calldata _hashedSignatures) external {
         address caller = _msgSender();
         uint256 length = _hashedSignatures.length;
         for (uint256 i = 0; i < length; ++i) {
             _cancelSignature(caller, _hashedSignatures[i]);
         }
     }
     ```

     Result: ACK.

3. The `_modifyTrade` function doesn't need return value, as `_trade` is passed by reference. Detail:

   - Change `_accept` function:

     ```
     function _accept(Trade memory _trade, address _caller) internal {
         _modifyTrade(_trade);
         // Use _trade instead of modifiedTrade below
     }
     ```

   - Change `_modifyTrade` function:

     ```
     function _modifyTrade(Trade memory _trade) internal view virtual {
         // Override
     }
     ```

   - In `DecentralandMarketplacePolygon.sol`, remove the return type and last return line of `_modifyTrade` function.

     Result: Changed.

4. As `accept` will be called frequently, it's better to split into `accept(Trade _trade)` and `acceptMany(Trade[] _trades)` (omit calldata for ease).

Result: ACK.

5. As above, for `cancelSignature`, it's better to split into `cancelSignature(single value)` and `cancelManySignatures(array)`.

   Result: ACK.

6. The first comment line of `cancelSignature` isn't accurate, as the param is an array.

   Result: Fixed.

DecentralandMarketplaceEthereum.sol

1. In `_transferERC721`, can pass `_asset.extra` directly(not need `decode` then `encode`):

   ```
   if (erc721.supportsInterface(erc721.verifyFingerprint.selector)) {
       if (!erc721.verifyFingerprint(_asset.value, _asset.extra)) {
           revert InvalidFingerprint();
       }
   }
   ```

   Result: Changed.

DecentralandMarketplacePolygon.sol

1. There isn't much need to define `sentLength`, `receivedLength`, `erc721` as they are used only once.

   Result: Changed.

2. For `_getFeesAndRoyalties`, as `_royaltyBeneficiaries` is passed by reference, so not need return it. Details:

   - In `_modifyTrade` function:

     ```
     (payFeeCollector, royaltyBeneficiariesCount) =
     _getFeesAndRoyalties(payFeeCollector, royaltyBeneficiariesCount,
     royaltyBeneficiaries, _trade.sent);
     (payFeeCollector, royaltyBeneficiariesCount) =
     _getFeesAndRoyalties(payFeeCollector, royaltyBeneficiariesCount,
     royaltyBeneficiaries, _trade.received);
     ```

   - for `_getFeesAndRoyalties` function

   ```
   1. Change the return type to "returns (bool, uint256)"
   2. Change the last line to "return (_payFeeCollector, _royaltyBeneficiariesCount);"
   ```

   Result: Changed.

3. For `_updateERC20sWithFees`, `_assets` is passed by reference, so not need return it. Details:

   - In `_modifyTrade` function:

```
_updateERC20sWithFees(_trade.sent, encodedFeeAndRoyaltyData);
_updateERC20sWithFees(_trade.received, encodedFeeAndRoyaltyData);
```

- In `_updateERC20sWithFees` function:

  Remove the return type and last return line.

Result: Changed.

4. In `_updateERC20sWithFees`, it's better to define an `asset` variable first and use it below.

   Result: Changed.

CouponManager.sol

1. In `updateAllowedCoupons`, the two params can change to `calldata`.

   Result: Changed.
2. The `applyCoupon` function not need be `virtual`.

   Result: Changed.
3. The first comment line of `cancelSignature` isn't accurate, as the param is an array.

   Result: Fixed.

CollectionDiscountCoupon.sol

1. The `_coupon` param can change to `calldata`.

   Result: Changed.

MarketplaceWithCouponManager.sol

1. The `couponManager` and `Coupon contracts` have more authority than they should. If one day one of these contracts is compromised, attacker may steal any users' assets for free. For example:

   > 1. User1 signs trade1
   > 2. Attacker call acceptWithCoupon(trade2, coupon2)

   The `coupon2` contract can change trade2's `signer` and `sent` to trade1's, and set trade2's `received` be empty, then user1's assets are stolen for free.

   So it's better to add a verification as below:

```
function acceptWithCoupon(Trade[] calldata _trades, Coupon[] calldata _coupons)
external whenNotPaused nonReentrant {
    _verifyTrade(_trades[i], caller);
    Trade memory appliedTrade = couponManager.applyCoupon(_trades[i], _coupons[i]);
    // Add a simple check, make sure signer doesn't get changed
    if (_trades[i].signer != appliedTrade.signer) {
        revert ..
    }
    _accept(appliedTrade, caller);
```

Or refactor as below(more strict):

> 1. Let `applyCoupon` only returns the changed value list(uint256[] memory).
> 2. In `acceptWithCoupon`, overwrite each `value` field of `trade.received` with the corresponding value upon.

Result: ACK. `We intend to use a MultiSig with multiple required signatures for any owner related operation so ownership compromises are prevented.`

Verifications.sol

1. As a trade can indeed be used multiple times(when `uses > 0`), maybe it's better to change `SignatureReuse` to `SignatureOveruse`.

   Result: Changed.

2. The bottom comment line of `_verifyExternalChecks` should be `..with the caller and value..`

   Result: Fixed.

3. If Trade can make sure all required checks are ahead of optional checks, the `_verifyExternalChecks` function can be simplified:

```
function _verifyExternalChecks(ExternalCheck[] calldata _externalChecks, address
_caller) .. {
    uint256 length = _externalChecks.length;
    for (uint256 i = 0; i < length; ++i) {
        ExternalCheck calldata externalCheck = _externalChecks[i];
        // deal selector and set success flag..
        if (externalCheck.required) {
            if (!success) {
                revert ExternalChecksFailed();
            }
        } else {
            if (success) {
                return;
            }
        }
    }

    if (!_externalChecks[length-1].required) {
        revert ExternalChecksFailed();
    }
}
```

Result: ACK.

FeeCollector.sol

1. All import lines and `is MarketplaceTypes` can be removed.

   Result: Fixed.

MarketplaceTypesHashing.sol

1. The import of `CommonTypes` can be removed.

   Result: Changed.

IRoyaltiesManager.sol

1. The import of `IERC721` and `is IERC721` can be removed, as `RoyaltiesManager` doesn't inherit `ERC721` actually.

   Result: Fixed.

IComposable.sol

1. It's better to add `view` for `verifyFingerprint` function.

   Result: Changed.

Recommendations

1. In `CollectionDiscountCoupon.sol`, it's worth considering whether `originalPrice - data.discount` should be `originalPrice > data.discount ? originalPrice - data.discount : 0`, to allow caller to use a bigger flat coupon.

   Result: ACK.

2. In `CollectionDiscountCoupon.sol`, an options is: Treat `data.root` as the contractAddress if `callerData.proofs.length == 0`. So if `_trade.sent` contains the same contractAddress, caller doesn't need provide `proofs` to call `MerkleProof.verify`. Detail:

   ```
   uint256 sentLength = _trade.sent.length;
   bool rootIsAddress = (callerData.proofs.length == 0);
   ```

```
if (sentLength == 0 || (!rootIsAddress && sentLength != callerData.proofs.length)) {
    revert InvalidSentOrProofsLength();
}
for (uint256 i = 0; i < sentLength; ++i) {
    ..
    if (rootIsAddress) {
        if (data.root != bytes32(bytes20(collectionAddress))) {
            revert InvalidProof(i);
        }
    } else {
        if (!MerkleProof.verify(..)) {
            revert InvalidProof(i);
        }
    }
}
```

Result: ACK.