ZooDAO / Mirror NFT Bridge

[WP-H1] Using only the address (originalCollectionAddress) is not sufficient to determine the identity of a unique NFT without the chainID.

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L138-L155

```
138 if (isReflection[collectionAddr]) {
139     //  NFT is reflection - burn
140
141     for (uint256 i = 0; i < tokenIds.length; i++) {
142         collection.burn(msg.sender, tokenIds[i]);
143     }
144
145     originalCollectionAddress = originalCollectionAddresses[collectionAddr];
146 } else {
147     // Is original NFT - lock NFT
148
149     for (uint256 i = 0; i < tokenIds.length; i++) {
150         collection.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
151     }
152
153     isOriginalChainForCollection[collectionAddr] = true;
154     originalCollectionAddress = collectionAddr;
155 }
```

There are NFT collections that share the same address across multiple networks:

- https://etherscan.io/address/0xc1248efe4cee8e2341bc736fcc634067c64a55a6
- https://polygonscan.com/token/0xc1248efe4cee8e2341bc736fcc634067c64a55a6

In such cases, the id of the collection should be a combination of networkId and address (e.g., `{networkId}:{address}`).

Otherwise, the NFT collections with the same address on different networks will be confused in the mirror bridge system.

[WP-M2] Return NFT to OriginalChain may fail when the owner is a smart contract

When unlocking the NFT on the original chain, `safeTransferFrom()` is used to facilitate the transfer at L205. However, the `owner` may have changed, and when the new address is a smart contract with no `onERC721Received()` method, the transfer will fail.

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L191-L231

```
191 function _reflect(
192     address originalCollectionAddr,
```

```
193     string memory name,
194     string memory symbol,
195     uint256[] memory tokenIds,
196     string[] memory tokenURIs,
197     address _owner
198 ) internal {
199     bool isOriginalChain = isOriginalChainForCollection[originalCollectionAddr];
200 
201     if (isOriginalChain) {
202         // Unlock NFT and return to owner
203 
204         for (uint256 i = 0; i < tokenIds.length; i++) {
206         }
207 
208         emit NFTReturned(originalCollectionAddr, tokenIds, _owner);
209     } else {
  ▶ @@ 210,229 @@
230     }
231 }
```

[WP-M3] Mirror.createReflection() cannot specify the address of the receiver for the NFT being bridged over to targetNetworkId

This is because at L157, `_owner` is hardcoded as `msg.sender`.

In the case where the owner of the NFT is a smart contract, this may result in users mistakenly transferring the NFT to an address that does not belong to the original owner, or getting stuck at L205 because the receiver at L205 (i.e., `_owner` in this case) cannot safely receive the NFT (as `onERC721Received()` is not implemented).

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L110-L162

```
110   function createReflection(
111     address collectionAddr,
112     uint256[] memory tokenIds,
113     uint16 targetNetworkId,
114     address payable _refundAddress,
115     address _zroPaymentAddress,
116     bytes memory _adapterParams
117   ) public payable {
118     require(isEligibleCollection[collectionAddr], 'Mirror: collection is not eligible');
119     require(tokenIds.length > 0, "Mirror: tokenIds weren't provided");
120     require(tokenIds.length <= reflectionAmountLimit, "Mirror: can't reflect more than limit");
121 
122     _deductFee();
123 
124     ReflectedNFT collection = ReflectedNFT(collectionAddr);
125 
126     string memory name = collection.name();
127     string memory symbol = collection.symbol();
128 
129     string[] memory tokenURIs = new string[](tokenIds.length);
130 
131     for (uint256 i = 0; i < tokenIds.length; i++) {
```

```
132       string memory tokenURI = collection.tokenURI(tokenIds[i]);
133       tokenURIs[i] = tokenURI;
134     }
135
136     address originalCollectionAddress;
137
138     if (isReflection[collectionAddr]) {
139       //  NFT is reflection - burn
140
141       for (uint256 i = 0; i < tokenIds.length; i++) {
142         collection.burn(msg.sender, tokenIds[i]);
143       }
144
145       originalCollectionAddress = originalCollectionAddresses[collectionAddr];
146     } else {
147       // Is original NFT - lock NFT
148
149       for (uint256 i = 0; i < tokenIds.length; i++) {
150         collection.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
151       }
152
153       isOriginalChainForCollection[collectionAddr] = true;
154       originalCollectionAddress = collectionAddr;
155     }
156
158
159     _lzSend(targetNetworkId, _payload, _refundAddress, _zroPaymentAddress, _adapterParams, msg.value
160
161     emit BridgeNFT(originalCollectionAddress, name, symbol, tokenIds, tokenURIs, msg.sender);
162   }
```

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L164-L231

```
164   /// @dev Function inherited from NonBlockingLzApp
165   /// @dev Called by lzReceive() that is triggered by LzEndpoint
166   /// @dev Calles _reflect() to finish bridge process
167   function _nonblockingLzReceive(uint16, bytes memory, uint64, bytes memory payload) internal virtual
168     (
169       address originalCollectionAddr,
170       string memory name,
171       string memory symbol,
172       uint256[] memory tokenIds,
173       string[] memory tokenURIs,
175     ) = abi.decode(payload, (address, string, string, uint256[], string[], address));
176
177     _reflect(originalCollectionAddr, name, symbol, tokenIds, tokenURIs, _owner);
178   }
179
180   /// @notice Function finishing bridge process
181   /// @notice Deploys ReflectedNFT contract if collection was bridged to current chain for the first
182   /// @notice Uses existing ReflectedNFT contract if collection was bridged to that chain before
183   /// @notice Mints NFT-reflection on ReflectedNFT contract
184   /// @notice Returns (unlocks) NFT to owner if current chain is original for bridged NFT
185   /// @param originalCollectionAddr Address of original collection on original chain as a unique iden
186   /// @param name name of original collection to mint ReflectedNFT if needed
187   /// @param symbol symbol of original collection to mint ReflectedNFT if needed
188   /// @param tokenIds Array of tokenIds of bridged NFTs to mint exact same tokens or to unlocks it
189   /// @param tokenURIs Array of tokenURIs of bridged NFTs to mint exact same tokens if needed
```

```
190   /// @param _owner Address to mint or return token to
191   function _reflect(
192     address originalCollectionAddr,
193     string memory name,
194     string memory symbol,
195     uint256[] memory tokenIds,
196     string[] memory tokenURIs,

198   ) internal {
199     bool isOriginalChain = isOriginalChainForCollection[originalCollectionAddr];
200
201     if (isOriginalChain) {
202       // Unlock NFT and return to owner
203
204       for (uint256 i = 0; i < tokenIds.length; i++) {

206       }
207
208       emit NFTReturned(originalCollectionAddr, tokenIds, _owner);
209     } else {
210       bool isThereReflectionContract = reflection[originalCollectionAddr] != address(0);
211
212       // Get ReflectedNFT address from storage (if exists) or deploy
213       address collectionAddr;
214
215       if (isThereReflectionContract) {
216         collectionAddr = reflection[originalCollectionAddr];
217       } else {
218         collectionAddr = _deployReflection(originalCollectionAddr, name, symbol);
219       }
220
221       // Make eligible to be able to bridge
222       isEligibleCollection[collectionAddr] = true;
223
224       // Mint NFT-reflections
225       for (uint256 i = 0; i < tokenIds.length; i++) {

227       }
228
229       emit NFTBridged(originalCollectionAddr, tokenIds, tokenURIs, _owner);
230     }
231   }
```

[WP-G4] Consider using Clones to reduce deployment and bridge costs.

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L95-L110

```
95    /// @notice Bridges NFT to target chain
96    /// @notice Locks original NFT on contract before bridge
97    /// @notice Burns reflection of NFT on bridge
98    /// @param collectionAddr A
99    /// @param tokenIds Array of tokenIds to bridge to target chain
100   /// @param targetNetworkId target network ID from LayerZero's ecosystem (different from chain ID)
101   /// @param _refundAddress Address to return excessive native tokens
102   /// @param _zroPaymentAddress Currently takes zero address, but left as parameter according to Layer
103   /// @param _adapterParams abi.encode(1, gasLimit) gasLimit for transaction on target chain
```

```
105   /// @dev _adapterParams`s gasLimit should be 300,000 for bridge of signle token to already deployed
106   /// @dev Original NFT collection is passed in message of bridge from any to any chain
107   /// @dev Original NFT collection address is used as a unique identifier at all chains
108   /// @dev In message provides name and symbols of bridged NFT collection to deploy exact same NFT co
109   /// @dev In message provides tokenIds and tokenURIs of bridged NFT to mint exact same NFTs on targe
110   function createReflection(
```

Deploying new contracts can be costly, especially when we consider the case that the deployment cost will incur overhead on LayerZero's crosschain message.

Clones is a library that can deploy cheap, minimal, non-upgradeable proxies.

Instead of deploying a new ReflectedNFT contract, creating a clone can be much cheaper in terms of gas.

[WP-N5] Unnecessary imports

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L5

```
5 import '@openzeppelin/contracts/access/Ownable.sol';
```

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L8

```
8 import '@layerzerolabs/solidity-examples/contracts/token/onft/ONFT721Core.sol';
```

[WP-N6] Consider adding collectionAddr (the NFT address on the target chain) to the NFTBridged event.

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L191-L231

```
191 function _reflect(
  ▶ @@ 192,197 @@
198 ) internal {
199     bool isOriginalChain = isOriginalChainForCollection[originalCollectionAddr];
200
201     if (isOriginalChain) {
  ▶ @@ 202,208 @@
209     } else {
210         bool isThereReflectionContract = reflection[originalCollectionAddr] =/= address(0);
211
212         // Get ReflectedNFT address from storage (if exists) or deploy
213         address collectionAddr;
214
215         if (isThereReflectionContract) {
216             collectionAddr = reflection[originalCollectionAddr];
217         } else {
218             collectionAddr = _deployReflection(originalCollectionAddr, name, symbol);
```

```
        }

        // Make eligible to be able to bridge
        isEligibleCollection[collectionAddr] = true;

        // Mint NFT-reflections
        for (uint256 i = 0; i < tokenIds.length; i++) {
            ReflectedNFT(collectionAddr).mint(_owner, tokenIds[i], tokenURIs[i]);
        }


    }
}
```

[WP-N7] Transferring NFT directly to Mirror can also trigger NFTReceived event

`onERC721Received()` may have forgotten to `require(operator == address(this), "...")`.

According to the comment in `NFTReceived`, it is expected to emit NFTReceived only when the Original NFT is locked into the `Mirror` contract.

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L257-L265

```
function onERC721Received(
    address operator,
    address from,
    uint256 tokenId,
    bytes calldata data
) external returns (bytes4) {
    emit NFTReceived(operator, from, tokenId, data);
    return IERC721Receiver.onERC721Received.selector;
}
```

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L33-L34

```
event NFTReceived(address operator, address from, uint256 tokenId, bytes data);
```

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L110-L162

```
function createReflection(
    address collectionAddr,
    uint256[] memory tokenIds,
    uint16 targetNetworkId,
    address payable _refundAddress,
    address _zroPaymentAddress,
    bytes memory _adapterParams
) public payable {
    require(isEligibleCollection[collectionAddr], 'Mirror: collection is not eligible');
```

```
119     require(tokenIds.length > 0, "Mirror: tokenIds weren't provided");
120     require(tokenIds.length ≤ reflectionAmountLimit, "Mirror: can't reflect more than limit");
121 
122     _deductFee();
123 
124     ReflectedNFT collection = ReflectedNFT(collectionAddr);
125 
126     string memory name = collection.name();
127     string memory symbol = collection.symbol();
128 
129     string[] memory tokenURIs = new string[](tokenIds.length);
130 
131     for (uint256 i = 0; i < tokenIds.length; i++) {
132         string memory tokenURI = collection.tokenURI(tokenIds[i]);
133         tokenURIs[i] = tokenURI;
134     }
135 
136     address originalCollectionAddress;
137 
138     if (isReflection[collectionAddr]) {
  ▶ @@ 139,145 @@
146     } else {
147         // Is original NFT - lock NFT
148 
149         for (uint256 i = 0; i < tokenIds.length; i++) {

151         }
152 
153         isOriginalChainForCollection[collectionAddr] = true;
154         originalCollectionAddress = collectionAddr;
155     }
156 
157     bytes memory _payload = abi.encode(originalCollectionAddress, name, symbol, tokenIds, tokenURIs, r
158 
159     _lzSend(targetNetworkId, _payload, _refundAddress, _zroPaymentAddress, _adapterParams, msg.value
160 
161     emit BridgeNFT(originalCollectionAddress, name, symbol, tokenIds, tokenURIs, msg.sender);
162 }
```