

ZooDao Mirror NFT Bridge Audit Report

Sep 24, 2023



Table of Contents

Summary	2
Overview	3
Issues	4
[WP-H1] Using only the address (originalCollectionAddress) is not sufficient to determine the identity of a unique NFT without the chainID.	4
[WP-M2] Return NFT to OriginalChain may fail when the owner is a smart contract	6
[WP-M3] Mirror.createReflection() cannot specify the address of the receiver for the NFT being bridged over to targetNetworkId	8
[WP-G4] Consider using Clones to reduce deployment and bridge costs.	13
[WP-N5] Unnecessary imports	15
[WP-N6] Consider adding collectionAddr (the NFT address on the target chain) to the NFTBridged event.	16
[WP-N7] Transferring NFT directly to Mirror can also trigger NFTReceived event	18
Appendix	21
Disclaimer	22



Summary

This report has been prepared for ZooDao Mirror NFT Bridge smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.



Overview

Project Summary

Project Name	ZooDao Mirror NFT Bridge
Codebase	https://github.com/ZooDAO-Project/mirror-nft-bridge
Commit	0d77a89d41b06920728fe02c7a195e83a0a06499
Language	Solidity

Audit Summary

Delivery Date	Sep 24, 2023
Audit Methodology	Static Analysis, Manual Review
Total Isssues	7



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

High

Issue Description

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

Medium

Issue Description

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,
          string[] memory tokenURIs,
196
          address _owner
197
      ) internal {
198
          bool isOriginalChain = isOriginalChainForCollection[originalCollectionAddr];
199
200
201
          if (isOriginalChain) {
              // Unlock NFT and return to owner
202
203
204
              for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
205
                  ReflectedNFT(originalCollectionAddr).safeTransferFrom(address(this),
      _owner, tokenIds[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

Medium

Issue Description

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

```
function createReflection(
110
              address collectionAddr,
111
              uint256[] memory tokenIds,
112
              uint16 targetNetworkId,
113
              address payable refundAddress,
114
              address _zroPaymentAddress,
115
              bytes memory _adapterParams
116
          ) public payable {
117
              require(isEligibleCollection[collectionAddr], 'Mirror: collection is not
118
      eligible');
119
              require(tokenIds.length > 0, "Mirror: tokenIds weren't provided");
              require(tokenIds.length <= reflectionAmountLimit, "Mirror: can't reflect</pre>
120
     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++) {</pre>
132
                  string memory tokenURI = collection.tokenURI(tokenIds[i]);
                  tokenURIs[i] = tokenURI;
133
134
              }
135
136
              address originalCollectionAddress;
137
138
              if (isReflection[collectionAddr]) {
139
                        NFT is reflection - burn
140
                  for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
141
142
                      collection.burn(msg.sender, tokenIds[i]);
                  }
143
144
                  originalCollectionAddress =
145
      originalCollectionAddresses[collectionAddr];
              } else {
146
147
                  // Is original NFT - lock NFT
148
149
                  for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
150
                      collection.safeTransferFrom(msg.sender, address(this),
      tokenIds[i]);
151
                  }
152
153
                  isOriginalChainForCollection[collectionAddr] = true;
                  originalCollectionAddress = collectionAddr;
154
155
              }
156
157
              bytes memory _payload = abi.encode(originalCollectionAddress, name,
      symbol, tokenIds, tokenURIs, msg.sender);
158
              _lzSend(targetNetworkId, _payload, _refundAddress, _zroPaymentAddress,
159
      _adapterParams, msg.value - feeAmount);
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

```
/// @dev Function inherited from NonBlockingLzApp
164
165
         /// @dev Called by LzReceive() that is triggered by LzEndpoint
         /// @dev Calles _reflect() to finish bridge process
166
         function nonblockingLzReceive(uint16, bytes memory, uint64, bytes memory
167
      payload) internal virtual override {
168
              (
                  address originalCollectionAddr,
169
170
                  string memory name,
171
                  string memory symbol,
172
                  uint256[] memory tokenIds,
                  string[] memory tokenURIs,
173
174
                  address owner
              ) = abi.decode(payload, (address, string, string, uint256[], string[],
175
     address));
176
177
              _reflect(originalCollectionAddr, name, symbol, tokenIds, tokenURIs,
      _owner);
          }
178
179
180
         /// @notice Function finishing bridge process
         /// @notice Deploys ReflectedNFT contract if collection was bridged to current
181
     chain for the first time
         /// @notice Uses existing ReflectedNFT contract if collection was bridged to
182
      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
         /// @param originalCollectionAddr Address of original collection on original
185
     chain as a unique identifier
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,
              string memory name,
193
              string memory symbol,
194
              uint256[] memory tokenIds,
195
```



```
196
              string[] memory tokenURIs,
197
              address _owner
198
          ) internal {
199
              bool isOriginalChain =
     isOriginalChainForCollection[originalCollectionAddr];
200
              if (isOriginalChain) {
201
                  // Unlock NFT and return to owner
202
203
                  for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
204
205
                      ReflectedNFT(originalCollectionAddr).safeTransferFrom(address(this),
      _owner, tokenIds[i]);
206
                  }
207
                  emit NFTReturned(originalCollectionAddr, tokenIds, owner);
208
209
              } else {
                  bool isThereReflectionContract = reflection[originalCollectionAddr] !=
210
     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 {
                      collectionAddr = _deployReflection(originalCollectionAddr, name,
218
     symbol);
                  }
219
220
                  // Make eligible to be able to bridge
221
                  isEligibleCollection[collectionAddr] = true;
222
223
224
                  // Mint NFT-reflections
225
                  for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
                      ReflectedNFT(collectionAddr).mint(_owner, tokenIds[i],
226
     tokenURIs[i]);
227
                  }
228
229
                  emit NFTBridged(originalCollectionAddr, tokenIds, tokenURIs, _owner);
230
             }
         }
231
```







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

Gas

Issue Description

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
         /// @notice Burns reflection of NFT on bridge
         /// @param collectionAddr A
98
         /// @param tokenIds Array of tokenIds to bridge to target chain
99
         /// @param targetNetworkId target network ID from LayerZero's ecosystem
100
     (different from chain ID)
         /// @param refundAddress Address to return excessive native tokens
101
102
         /// @param _zroPaymentAddress Currently takes zero address, but left as
     parameter according to LayerZero`s guidelines
         /// @param _adapterParams abi.encode(1, gasLimit) gasLimit for transaction on
103
     target chain
         /// @dev _adapterParams`s gasLimit should be 2,200,000 for bridge of single
104
     token to a new chain (chain where is no ReflectedNFT contract)
         /// @dev _adapterParams`s gasLimit should be 300,000 for bridge of signle
105
     token to already deployed ReflectedNFT contract
         /// @dev Original NFT collection is passed in message of bridge from any to
106
     any chain
         /// @dev Original NFT collection address is used as a unique identifier at all
107
     chains
         /// @dev In message provides name and symbols of bridged NFT collection to
108
     deploy exact same NFT contract on target chain
109
         /// @dev In message provides tokenIds and tokenURIs of bridged NFT to mint
     exact same NFTs on target chain
         function createReflection(
110
```

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

Issue Description

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';

Status

✓ Fixed



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

Issue Description

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

```
function reflect(
191
     @@ 192,197 @@
     ) internal {
198
199
          bool isOriginalChain = isOriginalChainForCollection[originalCollectionAddr];
200
201
         if (isOriginalChain) {
     @@ 202,208 @@
209
         } else {
              bool isThereReflectionContract = reflection[originalCollectionAddr] !=
210
     address(0);
211
212
              // Get ReflectedNFT address from storage (if exists) or deploy
              address collectionAddr;
213
214
215
              if (isThereReflectionContract) {
216
                  collectionAddr = reflection[originalCollectionAddr];
217
              } else {
                  collectionAddr = _deployReflection(originalCollectionAddr, name,
218
     symbol);
219
              }
220
221
              // Make eligible to be able to bridge
              isEligibleCollection[collectionAddr] = true;
222
223
224
             // Mint NFT-reflections
              for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
225
                  ReflectedNFT(collectionAddr).mint( owner, tokenIds[i], tokenURIs[i]);
226
              }
227
228
              emit NFTBridged(originalCollectionAddr, tokenIds, tokenURIs, owner);
229
230
         }
231
     }
```







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

Issue Description

```
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

```
257
     function onERC721Received(
258
         address operator,
259
         address from,
260
         uint256 tokenId,
         bytes calldata data
261
     ) external returns (bytes4) {
262
263
         emit NFTReceived(operator, from, tokenId, data);
         return IERC721Receiver.onERC721Received.selector;
264
265
     }
```

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

```
/// @dev Triggered on NFT being transfered from user to lock it on contract if
collection is original
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,
```



```
114
          address payable refundAddress,
          address _zroPaymentAddress,
115
116
          bytes memory _adapterParams
117
      ) public payable {
         require(isEligibleCollection[collectionAddr], 'Mirror: collection is not
118
     eligible');
         require(tokenIds.length > 0, "Mirror: tokenIds weren't provided");
119
120
          require(tokenIds.length <= reflectionAmountLimit, "Mirror: can't reflect more</pre>
     than limit");
121
122
         _deductFee();
123
124
         ReflectedNFT collection = ReflectedNFT(collectionAddr);
125
          string memory name = collection.name();
126
         string memory symbol = collection.symbol();
127
128
129
         string[] memory tokenURIs = new string[](tokenIds.length);
130
131
         for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
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
              for (uint256 i = 0; i < tokenIds.length; i++) {</pre>
149
                  collection.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
150
151
              }
152
153
              isOriginalChainForCollection[collectionAddr] = true;
154
              originalCollectionAddress = collectionAddr;
155
         }
156
157
          bytes memory _payload = abi.encode(originalCollectionAddress, name, symbol,
     tokenIds, tokenURIs, msg.sender);
158
```



```
159     _lzSend(targetNetworkId, _payload, _refundAddress, _zroPaymentAddress,
     _adapterParams, msg.value - feeAmount);
160
161     emit BridgeNFT(originalCollectionAddress, name, symbol, tokenIds, tokenURIs,
     msg.sender);
162 }
```





Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.



Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.