ZooDao Mirror NFT Bridge Audit Report

Sep 24, 2023

Table of Contents

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

```
if (isReflection[collectionAddr]) {
    //    NFT is reflection - burn

    for (uint256 i = 0; i < tokenIds.length; i++) {
        collection.burn(msg.sender, tokenIds[i]);
    }

    originalCollectionAddress = originalCollectionAddresses[collectionAddr];
} else {
    // Is original NFT - lock NFT

    for (uint256 i = 0; i < tokenIds.length; i++) {
        collection.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
    }

    isOriginalChainForCollection[collectionAddr] = true;
    originalCollectionAddress = collectionAddr;
}
```

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.

Status

✓ **Fixed**

[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,
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++) {
205              ReflectedNFT(originalCollectionAddr).safeTransferFrom(address(this),
     _owner, tokenIds[i]);
206          }
207
208          emit NFTReturned(originalCollectionAddr, tokenIds, _owner);
209      } else {
     @@ 210,229 @@
230      }
231  }
```

Status

✓ **Fixed**

[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

```
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 
```

```
        string[] memory tokenURIs = new string[](tokenIds.length);

        for (uint256 i = 0; i < tokenIds.length; i++) {
            string memory tokenURI = collection.tokenURI(tokenIds[i]);
            tokenURIs[i] = tokenURI;
        }

        address originalCollectionAddress;

        if (isReflection[collectionAddr]) {
            //    NFT is reflection - burn

            for (uint256 i = 0; i < tokenIds.length; i++) {
                collection.burn(msg.sender, tokenIds[i]);
            }

            originalCollectionAddress = originalCollectionAddresses[collectionAddr];
        } else {
            // Is original NFT - lock NFT

            for (uint256 i = 0; i < tokenIds.length; i++) {
                collection.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
            }

            isOriginalChainForCollection[collectionAddr] = true;
            originalCollectionAddress = collectionAddr;
        }

        bytes memory _payload = abi.encode(originalCollectionAddress, name, symbol, tokenIds, tokenURIs, msg.sender);

        _lzSend(targetNetworkId, _payload, _refundAddress, _zroPaymentAddress, _adapterParams, msg.value - feeAmount);

        emit BridgeNFT(originalCollectionAddress, name, symbol, tokenIds, tokenURIs, msg.sender);
    }
```

https://github.com/ZooDAO-Project/mirror-nft-bridge/blob/

90f87d7095afb7a8cf7e81d6f7965003a758f6b4/contracts/Mirror.sol#L164-L231

```
    /// @dev Function inherited from NonBlockingLzApp
    /// @dev Called by lzReceive() that is triggered by LzEndpoint
    /// @dev Calles _reflect() to finish bridge process
    function _nonblockingLzReceive(uint16, bytes memory, uint64, bytes memory payload) internal virtual override {
        (
            address originalCollectionAddr,
            string memory name,
            string memory symbol,
            uint256[] memory tokenIds,
            string[] memory tokenURIs,
            address _owner
        ) = abi.decode(payload, (address, string, string, uint256[], string[], address));

        _reflect(originalCollectionAddr, name, symbol, tokenIds, tokenURIs, _owner);
    }

    /// @notice Function finishing bridge process
    /// @notice Deploys ReflectedNFT contract if collection was bridged to current chain for the first time
    /// @notice Uses existing ReflectedNFT contract if collection was bridged to that chain before
    /// @notice Mints NFT-reflection on ReflectedNFT contract
    /// @notice Returns (unlocks) NFT to owner if current chain is original for bridged NFT
    /// @param originalCollectionAddr Address of original collection on original chain as a unique identifier
    /// @param name name of original collection to mint ReflectedNFT if needed
    /// @param symbol symbol of original collection to mint ReflectedNFT if needed
    /// @param tokenIds Array of tokenIds of bridged NFTs to mint exact same tokens or to unlocks it
    /// @param tokenURIs Array of tokenURIs of bridged NFTs to mint exact same tokens if needed
    /// @param _owner Address to mint or return token to
    function _reflect(
        address originalCollectionAddr,
        string memory name,
        string memory symbol,
        uint256[] memory tokenIds,
        string[] memory tokenURIs,
        address _owner
    ) internal {
        bool isOriginalChain = isOriginalChainForCollection[originalCollectionAddr];

        if (isOriginalChain) {
            // Unlock NFT and return to owner

            for (uint256 i = 0; i < tokenIds.length; i++) {
                ReflectedNFT(originalCollectionAddr).safeTransferFrom(address(this), _owner, tokenIds[i]);
            }

            emit NFTReturned(originalCollectionAddr, tokenIds, _owner);
        } else {
            bool isThereReflectionContract = reflection[originalCollectionAddr] != address(0);

            // Get ReflectedNFT address from storage (if exists) or deploy
            address collectionAddr;

            if (isThereReflectionContract) {
                collectionAddr = reflection[originalCollectionAddr];
            } else {
                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]);
            }

            emit NFTBridged(originalCollectionAddr, tokenIds, tokenURIs, _owner);
        }
    }
```

Status

✓ Fixed

[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
 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 LayerZero`s guidelines
103     /// @param _adapterParams abi.encode(1, gasLimit) gasLimit for transaction on target chain
104     /// @dev _adapterParams`s gasLimit should be 2,200,000 for bridge of single token to a new chain (chain where is no ReflectedNFT contract)
105     /// @dev _adapterParams`s gasLimit should be 300,000 for bridge of signle token to already deployed ReflectedNFT contract
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 contract on target chain
109     /// @dev In message provides tokenIds and tokenURIs of bridged NFT to mint exact same NFTs on target chain
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.

Status

✓ **Fixed**

[WP-N5] Unnecessary imports

Issue Description

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

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

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

```
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

```
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);
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++) {
226              ReflectedNFT(collectionAddr).mint(_owner, tokenIds[i], tokenURIs[i]);
227          }
228
229          emit NFTBridged(originalCollectionAddr, tokenIds, tokenURIs, _owner);
230      }
231  }
```

Status

✓ **Fixed**

[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

```
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

```
/// @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,
    address payable _refundAddress,
    address _zroPaymentAddress,
    bytes memory _adapterParams
) public payable {
    require(isEligibleCollection[collectionAddr], 'Mirror: collection is not eligible');
    require(tokenIds.length > 0, "Mirror: tokenIds weren't provided");
    require(tokenIds.length <= reflectionAmountLimit, "Mirror: can't reflect more than limit");

    _deductFee();

    ReflectedNFT collection = ReflectedNFT(collectionAddr);

    string memory name = collection.name();
    string memory symbol = collection.symbol();

    string[] memory tokenURIs = new string[](tokenIds.length);

    for (uint256 i = 0; i < tokenIds.length; i++) {
        string memory tokenURI = collection.tokenURI(tokenIds[i]);
        tokenURIs[i] = tokenURI;
    }

    address originalCollectionAddress;

    if (isReflection[collectionAddr]) {
@@ 139,145 @@
    } else {
        // Is original NFT - lock NFT

        for (uint256 i = 0; i < tokenIds.length; i++) {
            collection.safeTransferFrom(msg.sender, address(this), tokenIds[i]);
        }

        isOriginalChainForCollection[collectionAddr] = true;
        originalCollectionAddress = collectionAddr;
    }

    bytes memory _payload = abi.encode(originalCollectionAddress, name, symbol, tokenIds, tokenURIs, msg.sender);

    _lzSend(targetNetworkId, _payload, _refundAddress, _zroPaymentAddress,
_adapterParams, msg.value - feeAmount);

    emit BridgeNFT(originalCollectionAddress, name, symbol, tokenIds, tokenURIs,
msg.sender);
}
```

Status

✓ **Fixed**

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