Oct 28, 2024

ZAN

Security Assessment MintRichNFT

—

Professional Service

Table of Contents

1. **Overview**
    - 1.1. Executive Summary
    - 1.2. Project Summary
    - 1.3. Assessment Summary
    - 1.4. Assessment Scope
2. **Checklist**
3. **Findings**
    - M-01: Use _safeMint/safeTransferFrom Instead of _mint/transferFrom for ERC721 Token Minting or Transferring
    - L-02: ERC404 Is Not Compatible With IERC721
    - I-03: Avoid Locking Funds in the Contract
    - I-04: No Check of Address Params with Zero Address
    - I-05: Function Visibility Can Be External
    - I-06: Prefer uint256
    - I-07: Floating Pragma
    - I-08: Unchecked Input Parameter collectionType
    - I-09: Streamlined Token Transfer Procedure
    - I-10: Redundant Zero Address Check
    - I-11: Unused Imported Contract
4. **Disclaimer**
5. **Appendix**

1. Overview

1.1. Executive Summary

MintRichNFT is a project for creating NFT contracts. This report has been prepared for NFTScan to discover issues and vulnerabilities in the source code of this project as well as any contract dependencies that were not part of an officially recognized library.

Conducted by Static Analysis, Formal Verification and Manual Review, we have identified **1 Medium and 5 Informational issues** in commit fa6e0e4, **1 Low and 3 Informational issues** in commit e77c499 and **1 Informational issue** in commit 8441f79.

The project team **has partially resolved the security vulnerability described in M-01** in commit 6b8984c after assessing the potential harm of the issue and the gas consumption. About the issue described in L-02, the project team has confirmed that the present implementation fulfills the business logic requirements. Therefore, they decided to keep no change. For details on the resolution of the informational issues, please refer to the *Alleviation* section.

1.2. Project Summary

Project Name	MintRichNFT
Platform	Mint/Ethereum
Language	Solidity
Codebase	Audit 1: • https://github.com/mintrich/MintRich-Contracts/tree/fa6e0e4b7b2431df3fc06833ab213a76dc3c63d7 Audit 2: • https://github.com/mintrich/MintRich-Contracts/tree/3a32f603449307dcd42b0fff7bc5a839017fd8c3 Audit 3: • https://github.com/mintrich/MintRich-Contracts/tree/e77c49951409790855ccb5b4ed0fdc1fd6e6951b Audit 4: • https://github.com/mintrich/MintRich-Contracts/tree/6b8984cd1b773f9040e9a5eabfefc7084b8b1e8a Audit 5: • https://github.com/mintrich/MintRich-Contracts/tree/8441f79b625bb428a3697da11ecdad72de2dd421 Final Audit: • https://github.com/mintrich/MintRich-Contracts/tree/cfcd2f9dd2b3ac2dc67e45a65dc7ab8ea52faf18

1.3. Assessment Summary

Delivery Date	Oct 28, 2024
Audit Methodology	Static Analysis, Formal Verification, Manual Review

1.4. Assessment Scope

ID	File	File Hash
1	/src/rich/MintRichNFTContract.sol	591beb05d486b1b48a4c08fe59cef928
2	/src/metadata/MetadataRenderer.sol	341627149ab32d17c6063b313d235568
3	/src/libs/MintRichPriceLib.sol	febb63323128b3eb8c7b6c4697c96f39
4	/src/metadata/IMetadataRenderer.sol	9c0ba7b59a4dba71c26cf2f01e7daf5
5	/src/MintRichNFTFactoryContract.sol	182369bcb0abfa930860454133cfb03a
6	/src/rich/MintRichCommonStorage.sol	caf1dcb58023e0c86e4b46f70d2e8937
7	/src/rich/ERC721AQueryableUpgradeable.sol	9e9cf260c4fd98db99e9d65a56d48f0c
8	/src/rich/lib/ERC721Events.sol (commit e77c499)	273dc5c6e50951d4353940a1b833ada0
9	/src/rich/erc404/ERC404.sol (commit e77c499)	4fc639efd614ee0cffbe47b357a9de53
10	/src/rich/lib/ERC20Events.sol (commit e77c499)	eb7dc726d350391c3a964ff54af8bc2f
11	/src/rich/erc404/IERC404.sol (commit e77c499)	11685d9e93f554a8ff666b2a0dc7684f
12	/src/rich/MintRich404NFTContract.sol (commit e77c499)	b4ed91e6d393c5964515304c975498fe
13	/src/rich/lib/DoubleEndedQueueUint.sol (commit e77c499)	885c53fbc7882bd39ef7a1ab326af9e
14	/src/rich/MintRich20NFTContract.sol (commit 8441f79)	f8a6583a6c30360fe7a2b094d23850b3

2. Checklist

2.1. Code Security

Reentrancy	DelegateCall	Integer Overflow
Input Validation	Unchecked this.call	Frozen Money
Arbitrary External Call	Unchecked Owner Transfer	Do-while Continue
Right-To-Left-Override Character	Unauthenticated Storage Access	Risk For Weak Randomness
TxOrigin	Missing Checks for Return Values	Diamond Inheritance
ThisBalance	VarType Deduction	Array Length Manipulation
Uninitialized Variable	Shadow Variable	Divide Before Multiply
Affected by Compiler Bug		

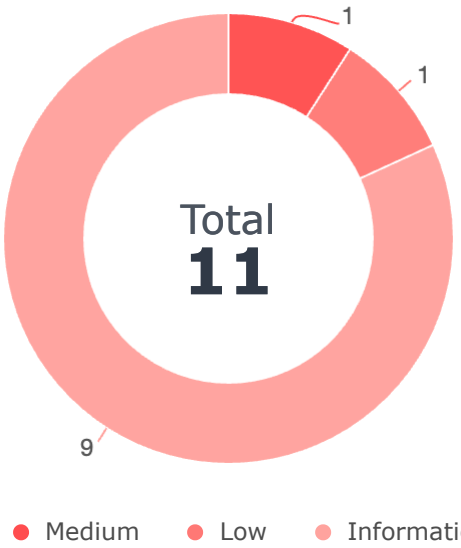
2.2. Optimization Suggestion

Compiler Version	Improper State Variable Modification
Function Visibility	Deprecated Function
Externally Controlled Variables	Code Style
Constant Specific	Event Specific
Return Value Unspecified	Inexistent Error Message
State Variable Defined Without Storage Location	Import Issue
Compare With Timestamp/Block Number/Blockhash	Constructor in Base Contract Not Implemented
Delete Struct Containing the Mapping Type	Usage of '=+'
Paths in the Modifier Not End with "_" or Revert	Non-payable Public Functions Use msg.value
Lack of SafeMath	Compiler Error/Warning
Tautology Issue	Loop Depends on Array Length
Redundant/Duplicated/Dead Code	Code Complexity/Code Inefficiency
Undeclared Resource	Optimizable Return Statement
Unused Resource	

2.3. Business Security

The Code Implementation is Consistent With Comments, Project White Papers and Other Materials
Permission Check
Address Check

3. Findings



Code Security

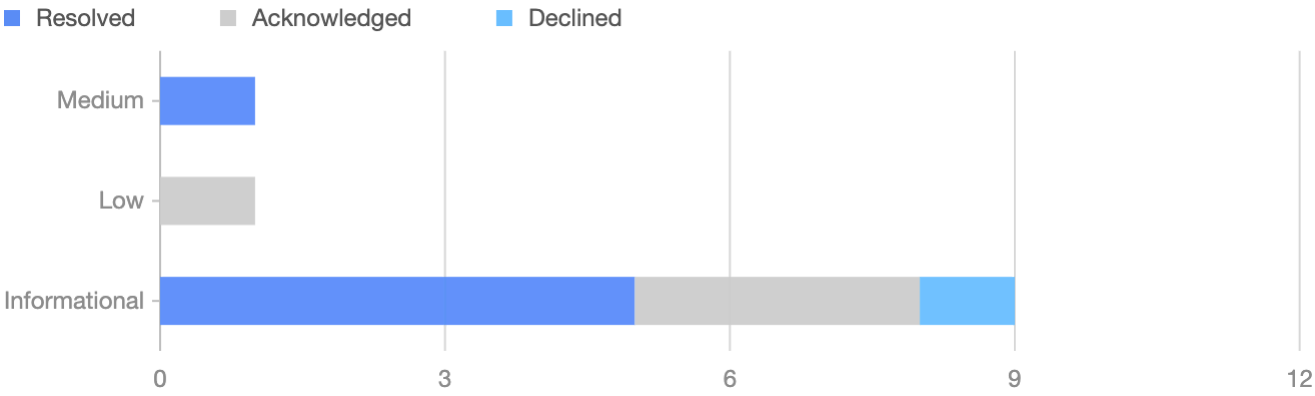
Medium: 1, Low: 1

Optimization Suggestion

Informational: 9

Business Security

no vulnerabilities.



ID	Title	Category	Severity	Status
M-01	Use _safeMint/safeTransferFrom Instead of _mint/transferFrom for ERC721 Token Minting or Transferring	Code Security	Medium	Resolved
L-02	ERC404 Is Not Compatible With IERC721	Code Security	Low	Acknowledged
I-03	Avoid Locking Funds in the Contract	Optimization Suggestion	Informational	Declined
I-04	No Check of Address Params with Zero Address	Optimization Suggestion	Informational	Resolved
I-05	Function Visibility Can Be External	Optimization Suggestion	Informational	Resolved
I-06	Prefer uint256	Optimization Suggestion	Informational	Acknowledged
I-07	Floating Pragma	Optimization Suggestion	Informational	Acknowledged
I-08	Unchecked Input Parameter collectionType	Optimization Suggestion	Informational	Resolved

ID	Title	Category	Severity	Status
I-09	Streamlined Token Transfer Procedure	Optimization Suggestion	Informational	Resolved
I-10	Redundant Zero Address Check	Optimization Suggestion	Informational	Acknowledged
I-11	Unused Imported Contract	Optimization Suggestion	Informational	Resolved

M-01: Use _safeMint/safeTransferFrom Instead of _mint/transferFrom for ERC721 Token Minting or Transferring

Medium: Code Security

File Location:
/src/MintRichNFTFactoryContract.sol:52
/src/rich/MintRichNFTContract.sol:106,112

Description

It is recommended to use `_safeMint` / `safeTransferFrom` instead of the `_mint` / `transferFrom` functions to mint or transfer ERC721 tokens, as the former ensures that the recipient is an EOA or a contract that implements the `onERC721Received` callback function. This helps to avoid the scenario where the ERC721 tokens are locked in a particular contract.

/src/MintRichNFTFactoryContract.sol

```
34  function createRichCollection(
35      bytes32 collectionId,
36      string calldata name,
37      string calldata symbol,
38      bytes32 packedData,
39      bytes calldata information
40  ) external {
41      checkCaller(collectionId);
42      address collection = Clones.cloneDeterministic
        (implementationAddress, collectionId);
43
44      (bool success, bytes memory returnData) = collection.call(abi.
        encodeCall(
45          MintRichNFTContract.initialize, (name, symbol, packedData,
            information)));
46      if (!success) {
47          assembly {
48              revert(add(returnData, 32), mload(returnData))
49          }
50      }
51
52      _mint(msg.sender, uint256(uint160(collection)));
53      emit MintRichCollectionCreated(msg.sender, collection, collectionId,
        name, symbol);
54  }
```

/src/rich/MintRichNFTContract.sol

```
 97  function buyNFTs(uint256 amount) internal {
 98      address buyer = msg.sender;
 99      uint256 mintAmount = amount;
100
101      if (!bank.empty()) {
102          uint256 withdrawAmount = Math.min(amount, bank.length());
103          for (uint256 i = 0; i < withdrawAmount; ++i) {
104              uint256 tokenId = uint256(bank.popBack());
105              _approve(buyer, tokenId);
106              transferFrom(address(this), buyer, tokenId);
107          }
108          mintAmount = amount - withdrawAmount;
109      }
110
111      if (mintAmount > 0) {
112          _mint(buyer, mintAmount);
113      }
114  }
```

Recommendation

It is recommended to use `_safeMint`/`safeTransferFrom` instead of the `_mint`/`transferFrom` functions to mint or transfer ERC721 tokens.

Alleviation

Acknowledged and partially resolved in commit 6b8984c. To save gas, the project team decided not to make any modifications to function `buyNFTs`.

L-02: ERC404 Is Not Compatible With IERC721

Low: Code Security

File Location: /src/rich/erc404/ERC404.sol (commit e77c499):462

Description

The `ERC404` contract does not fully adhere to the interface specified by the ERC-721 standard. It employs `erc721BalanceOf` rather than the standard `balanceOf` function to return the number of ERC-721 tokens in user's account. Consequently, it should not assert compatibility with `IERC721` in its `supportsInterface` function, as this may disrupt the standard interface detection mechanism outlined in ERC-165.

/src/rich/erc404/ERC404.sol (commit e77c499)

```
456   function supportsInterface(
457       bytes4 interfaceId
458   ) public view virtual returns (bool) {
459       return
460           interfaceId == type(IERC404).interfaceId ||
461           interfaceId == type(IERC165).interfaceId ||
462           interfaceId == type(IERC721).interfaceId ||
463           interfaceId == type(IERC20).interfaceId
464           ;
465   }
```

Recommendation

Do not claim that the `ERC404` contract implements the interface of the ERC-721 standard in the `supportsInterface` function.

Alleviation

The project team acknowledged the issue and decided to keep no change, as the present implementation fulfills the business logic requirements.

I-03: Avoid Locking Funds in the Contract

Informational: Optimization Suggestion

File Location: /src/rich/MintRichNFTContract.sol:164

Description

In the `processSaleClosed` function, 10% of the `totalBalance` (calculated as `share`) is transferred to the `MINT_RICH_RECIPIENT`, and the remaining 90% (calculated as `bids`) is approved to the `MINTSWAP_NFT_MARKETPLACE` address to execute bidding operation. Both `share` and `bids` are calculated based on their respective proportions of the `totalBalance`.

Due to Solidity's characteristic of rounding down during division, the sum of `share` and `bids` may not precisely equal the `totalBalance`, which can result in a small portion of the funds being frozen in the contract.

/src/rich/MintRichNFTContract.sol

```
156  function processSaleClosed() external nonReentrant {
157      require(msg.sender == MINT_RICH_ADMIN, "Only admin can process");
158      require(salePhase == SalePhase.CLOSED, "Sale not closed");
159
160      uint256 totalBalance = saleBalance();
161      uint256 share = (totalBalance * MINT_RICH_SHARE_POINTS) /
         BASIS_POINTS;
162      MINT_RICH_RECIPIENT.sendValue(share);
163
164      uint256 bids = (totalBalance * MINT_RICH_BIDS_POINTS) / BASIS_POINTS;
165      Address.functionCallWithValue(WETH9, abi.encodeWithSelector
         (WETH9_DEPOSIT_SELECTOR), bids);
166      IERC20(WETH9).approve(MINTSWAP_NFT_MARKETPLACE, bids);
167
168      Address.functionCall(MINTSWAP_NFT_MARKETPLACE, abi.encodeWithSelector
         (
169          MINTSWAP_BIDS_SELECTOR,
170          address(this),
171          uint64(MAX_SUPPLY),
172          uint128(bids / MAX_SUPPLY),
173          MINTSWAP_BIDS_EXPIRATION_TIME,
174          WETH9));
175
176      Address.functionCall(factoryAddress, abi.encodeWithSelector
         (FACTORY_BURN_SELECTOR));
177  }
```

Recommendation

It is recommended to calculate the value of `bids` directly by subtracting `share` from `totalBalance`.

Alleviation

The project team has decided not to make any changes regarding this issue. After testing, the project team calculated that the `totalBalance` is exactly 10 ETH, so there is no precision loss while calculating `bids`.

I-04: No Check of Address Params with Zero Address

Informational: Optimization Suggestion

File Location: /src/MintRichNFTFactoryContract.sol:26

Description

The input parameter of the address type in the function does not use the zero address for verification.

/src/MintRichNFTFactoryContract.sol

```
24      }
25
26      function initialize(address _implementationAddress) initializer
        external {
27          __ERC721_init("MintRich Owner", "MROwner");
28          __Ownable_init(_msgSender());
```

Recommendation

It is recommended to perform zero address verification on the input parameters of the address type.

Alleviation

Resolved in commit 6b8984c.

I-05: Function Visibility Can Be External

Informational: Optimization Suggestion

File Location:
/src/MintRichNFTFactoryContract.sol:56
/src/rich/MintRichNFTContract.sol:71

Description

Functions that are not called should be declared as external.

/src/MintRichNFTFactoryContract.sol

```
54      }
55
56      function predictDeterministicAddress(bytes32 collectionId) public
        view returns (address) {
57          return Clones.predictDeterministicAddress(implementationAddress,
            collectionId, address(this));
58      }
```

/src/rich/MintRichNFTContract.sol

```
69      }
70
71      function owner() public view returns (address) {
72          if (salePhase == SalePhase.CLOSED) {
73              return address(0);
```

Recommendation

Functions that are not called in the contract should be declared as external.

Alleviation

Resolved in commit 6b8984c.

I-06: Prefer uint256

Informational: Optimization Suggestion

File Location: /src/rich/MintRichCommonStorage.sol:33

Description

It is recommended to use uint256 type to avoid gas overhead caused by 32 bytes padding.

/src/rich/MintRichCommonStorage.sol

```
31      uint8 internal constant IMAGE_TYPE_MULIT = 1;
32
33      uint8 public imageType;
34      string public baseURI;
35
```

Recommendation

It is recommended to use uint256 type to avoid gas overhead caused by 32 bytes padding.

Alleviation

The project team acknowledged the issue and decided to keep no change.

I-07: Floating Pragma

Informational: Optimization Suggestion

File Location:
/src/MintRichNFTFactoryContract.sol:2
/src/libs/MintRichPriceLib.sol:2
/src/metadata/IMetadataRenderer.sol:2
/src/metadata/MetadataRenderer.sol:2
/src/rich/ERC721AQueryableUpgradeable.sol:5
/src/rich/MintRichCommonStorage.sol:2
/src/rich/MintRichNFTContract.sol:2

Description

Contracts should be deployed with fixed compiler version which has been tested thoroughly or make sure to lock the contract compiler version in the project configuration. Locked compiler version ensures that contracts will not be compiled by untested compiler version.

/src/MintRichNFTFactoryContract.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.8.20;
3
4  import "@openzeppelin/contracts-upgradeable/token/ERC721/
   ERC721Upgradeable.sol";
```

/src/libs/MintRichPriceLib.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.8.20;
3
4  import { UD60x18, ud, uUNIT, mul, sqrt} from "prb-math/UD60x18.sol";
```

/src/metadata/IMetadataRenderer.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.20;
3
4  interface IMetadataRenderer {
```

/src/metadata/MetadataRenderer.sol

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.20;
3
4  import "@openzeppelin/contracts/access/Ownable.sol";
```

/src/rich/ERC721AQueryableUpgradeable.sol

```
3  // Creator: Chiru Labs
4
5  pragma solidity ^0.8.20;
6
7  import 'lib/ERC721A-Upgradeable/contracts/extensions/
   IERC721AQueryableUpgradeable.sol';
```

/src/rich/MintRichCommonStorage.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.8.20;
3
4  import "@openzeppelin/contracts/utils/structs/DoubleEndedQueue.sol";
```

/src/rich/MintRichNFTContract.sol

```
1  // SPDX-License-Identifier: UNLICENSED
2  pragma solidity ^0.8.20;
3
4  import "./ERC721AQueryableUpgradeable.sol";
```

Recommendation

Use a fixed compiler version, and consider whether the bugs in the selected compiler version (https://github.com/ethereum/solidity/releases) will affect the contract.

Alleviation

The project team acknowledged the issue and decided to keep no change.

I-08: Unchecked Input Parameter collectionType

Informational: Optimization Suggestion

File Location: /src/MintRichNFTFactoryContract.sol (commit e77c499):50

Description

In the contract `MintRichNFTFactoryContract`, the `createRichCollection` function is called to create a collection. However, this function does not validate whether the input parameter `collectionType` is a valid collection type, which means that the `implementationTypes[collectionType]` may return a zero address. As a result, the minimal proxy (address of created collection contract) created through the `Clones` library points to an implementation contract that is a zero address.

/src/MintRichNFTFactoryContract.sol (commit e77c499)

```
41  function createRichCollection(
42      bytes32 collectionId,
43      uint256 collectionType,
44      string calldata name,
45      string calldata symbol,
46      bytes32 packedData,
47      bytes calldata information
48  ) external {
49      checkCaller(collectionId);
50      address collection = Clones.cloneDeterministic(implementationTypes
        [collectionType], collectionId);
51
52      (bool success, bytes memory returnData) = collection.call(abi.
        encodeCall(
53          MintRichNFTContract.initialize, (name, symbol, packedData,
            information)));
54      if (!success) {
55          assembly {
56              revert(add(returnData, 32), mload(returnData))
57          }
58      }
```

Recommendation

It is recommended to add a check for `implementationTypes[collectionType] != address(0)`.

Alleviation

Resolved in commit 6b8984c.

I-09: Streamlined Token Transfer Procedure

Informational: Optimization Suggestion

File Location: /src/rich/MintRich404NFTContract.sol (commit e77c499):97,98

Description

The `buyNFTs` function within the `MintRich404NFTContract` facilitates both ERC-20 and ERC-721 transfers from the contract to the purchaser, provided there are remaining tokens in the "bank". It is unnecessary to initially set the buyer's allowance over the contract's tokens through the `_erc20Approve` function, followed by executing the token transfers via the `transferFrom` function. Instead, directly invoking the internal `_transferERC20WithERC721` function, which bypasses the token allowance verification, is a more succinct and logical approach.

/src/rich/MintRich404NFTContract.sol (commit e77c499)

```
 89  function buyNFTs(uint256 amount) internal {
 90      address buyer = msg.sender;
 91      uint256 mintAmount = amount;
 92
 93      if (bank404 > 0) {
 94          uint256 withdrawAmount = Math.min(amount, bank404);
 95          uint256 amount404 = withdrawAmount * units;
 96
 97          _erc20Approve(buyer, amount404);
 98          transferFrom(address(this), buyer, amount404);
 99
100          bank404 -= withdrawAmount;
101          mintAmount = amount - withdrawAmount;
102      }
103
104      if (mintAmount > 0) {
105          _mintERC20(buyer, mintAmount * units);
106      }
107  }
```

Recommendation

Remove the redundant `_erc20Approove` function from the `ERC404` contract and directly invoke the `_transferERC20WithERC721` function within the `buyNFTs` function to execute token transfers. This approach streamlines the process by eliminating unnecessary steps.

Alleviation

Resolved in commit 6b8984c.

I-10: Redundant Zero Address Check

Informational: Optimization Suggestion

File Location:
/src/rich/MintRich404NFTContract.sol (commit e77c499):192
/src/rich/erc404/ERC404.sol (commit e77c499):104,105,106

Description

In the `tokenURI` function, the `ownerOf` function is called to check whether the specified `tokenId` exists. Since the `ownerOf` function already verifies the existence of the `tokenId`, there is no need to check it again in the `tokenURI` function.

/src/rich/MintRich404NFTContract.sol (commit e77c499)

```
189  function tokenURI(uint256 tokenId) public view override
190      returns (string memory)
191  {
192      require(ownerOf(tokenId) != address(0), "Token not exist");
```

/src/rich/erc404/ERC404.sol (commit e77c499)

```
 95  function ownerOf(
 96      uint256 id_
 97  ) public view virtual returns (address erc721Owner) {
 98      erc721Owner = _getOwnerOf(id_);
 99
100      if (!_isValidTokenId(id_)) {
101          revert InvalidTokenId();
102      }
103      // If the id_ is beyond the range of minted tokens, is 0, or the
         token is not owned by anyone, revert.
104      if (erc721Owner == address(0)) {
105          revert NotFound();
106      }
107  }
```

Recommendation

It is recommended to remove the unnecessary zero address check in function `tokenURI`.

Alleviation

The project team acknowledged the issue and decided to keep no change.

I-11: Unused Imported Contract

Informational: Optimization Suggestion

File Location: /src/rich/MintRich20NFTContract.sol (commit 8441f79):8,9

Description

Unused imported contract will increase gas consumption.

/src/rich/MintRich20NFTContract.sol (commit 8441f79)

```
6   import "@openzeppelin/contracts/utils/Address.sol";
7   import "@openzeppelin/contracts/utils/math/Math.sol";
8   import "@openzeppelin/contracts/utils/Base64.sol";
9   import "@openzeppelin/contracts/utils/Strings.sol";
10  import "@openzeppelin/contracts/utils/cryptography/ECDSA.sol";
```

Recommendation

It is recommended to delete unused imported contracts.

Alleviation

Resolved in commit cfcd2f9.

4. Disclaimer

No description, statement, recommendation or conclusion in this report shall be construed as endorsement, affirmation or confirmation of the project. The security assessment is limited to the scope of work as stipulated in the Statement of Work.

This report is prepared in response to source code, and based on the attacks and vulnerabilities in the source code that already existed or occurred before the date of this report, excluding any new attacks or vulnerabilities that exist or occur after the date of this report. The security assessment are solely based on the documents and materials provided by the customer, and the customer represents and warrants documents and materials are true, accurate and complete.

CONSULTANT DOES NOT MAKE AND HEREBY DISCLAIMS ANY REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY OR OTHERWISE, REGARDING THE SERVICES, DELIVERABLES, OR ANY OTHER MATTER PERTAINING TO THIS REPORT.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR AND HEREBY DISCLAIMS MERCHANTABILITY, FITNESS FOR PURPOSE, TITLE, NON-INFRINGEMENT OR NON-APPROPRIATION OF INTELLECTUAL PROPERTY RIGHTS OF A THIRD PARTY, SATISFACTORY QUALITY, ACCURACY, QUALITY, COMPLETENESS, TIMELINESS, RESPONSIVENESS, OR PRODUCTIVITY OF THE SERVICES OR DELIVERABLES.

CONSULTANT EXCLUDES ANY WARRANTY THAT THE SERVICES AND DELIVERABLES WILL BE UNINTERRUPTED, ERROR FREE, FREE OF SECURITY DEFECTS OR HARMFUL COMPONENTS, REVEAL ALL SECURITY VULNERABILITIES, OR THAT ANY DATA WILL NOT BE LOST OR CORRUPTED.

CONSULTANT SHALL NOT BE RESPONSIBLE FOR (A) ANY REPRESENTATIONS MADE BY ANY PERSON REGARDING THE SUFFICIENCY OR SUITABILITY OF SERVICES AND DELIVERABLES IN ANY ACTUAL APPLICATION, OR (B) WHETHER ANY SUCH USE WOULD VIOLATE OR INFRINGE THE APPLICABLE LAWS, OR (C) REVIEWING THE CUSTOMER MATERIALS FOR ACCURACY.

5. Appendix

5.1 Visibility

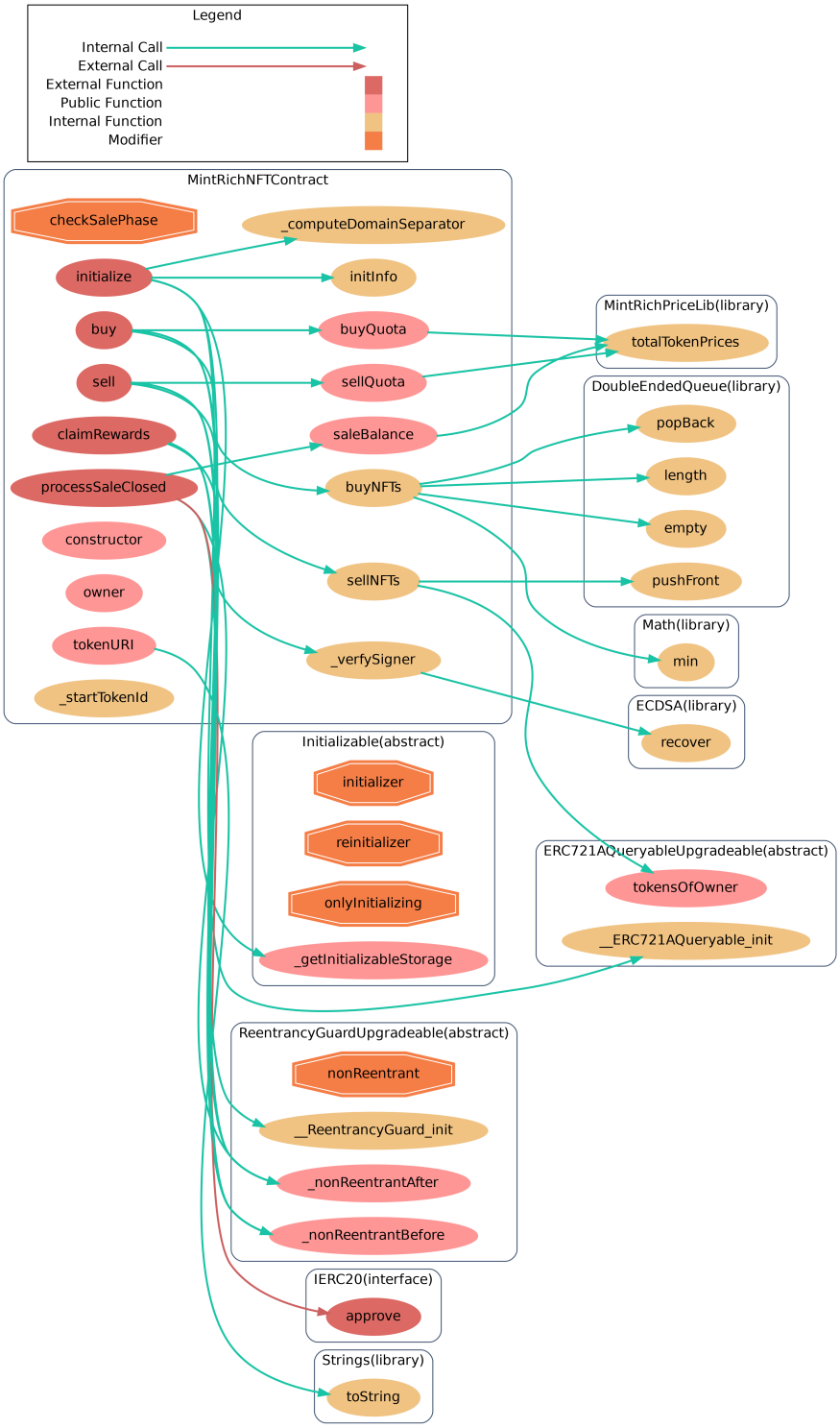
Contract	FuncName	Visibility	Mutability	Modifiers
MintRichNFTContract	_CTOR_	public	Y	
MintRichNFTContract	initialize	external	Y	initializerERC721A, initializer
MintRichNFTContract	initInfo	internal	N	
MintRichNFTContract	_startTokenId	internal	N	
MintRichNFTContract	owner	public	N	
MintRichNFTContract	buy	external	Y	nonReentrant, checkSalePhase
MintRichNFTContract	buyNFTs	internal	N	
MintRichNFTContract	sell	external	Y	nonReentrant, checkSalePhase
MintRichNFTContract	sellNFTs	internal	N	
MintRichNFTContract	buyQuota	public	N	
MintRichNFTContract	sellQuota	public	N	
MintRichNFTContract	saleBalance	public	N	
MintRichNFTContract	processSaleClosed	external	Y	nonReentrant
MintRichNFTContract	claimRewards	external	Y	nonReentrant
MintRichNFTContract	tokenURI	public	N	

Contract	FuncName	Visibility	Mutability	Modifiers
MintRichNFTContract	_verfySigner	internal	N	
MintRichNFTContract	_computeDomainSeparator	internal	N	
MintRichNFTFactoryContract	_CTOR_	public	Y	
MintRichNFTFactoryContract	initialize	external	Y	initializer
MintRichNFTFactoryContract	createRichCollection	external	Y	
MintRichNFTFactoryContract	predictDeterministicAddress	public	N	
MintRichNFTFactoryContract	checkCaller	internal	N	
MintRichNFTFactoryContract	_authorizeUpgrade	internal	N	onlyOwner
MintRichNFTFactoryContract	tokenURI	public	N	
MintRichNFTFactoryContract	setMetadataRenderer	external	Y	onlyOwner
MintRichNFTFactoryContract	setImplementationAddress	external	Y	onlyOwner
MintRichNFTFactoryContract	burnToken	external	Y	
MetadataRenderer	_CTOR_	public	Y	
MetadataRenderer	tokenURI	external	N	
MetadataRenderer	tokenURIJSON	public	N	
MetadataRenderer	setName	external	Y	onlyOwner
MetadataRenderer	setDescription	external	Y	onlyOwner

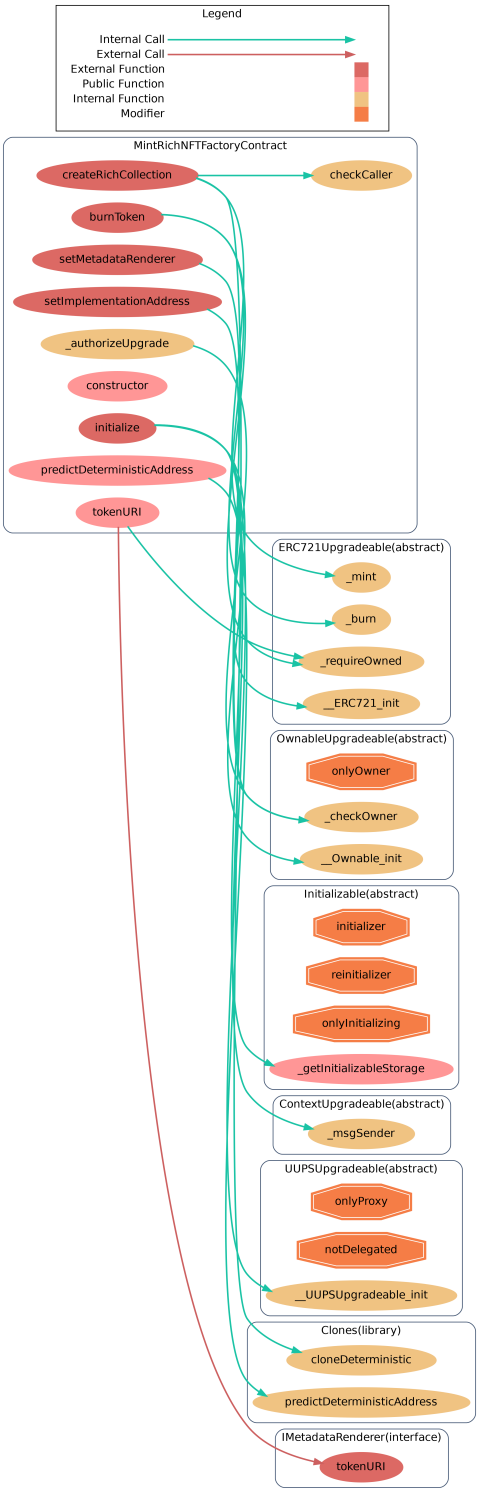
5. Appendix

5.2 Call Graph

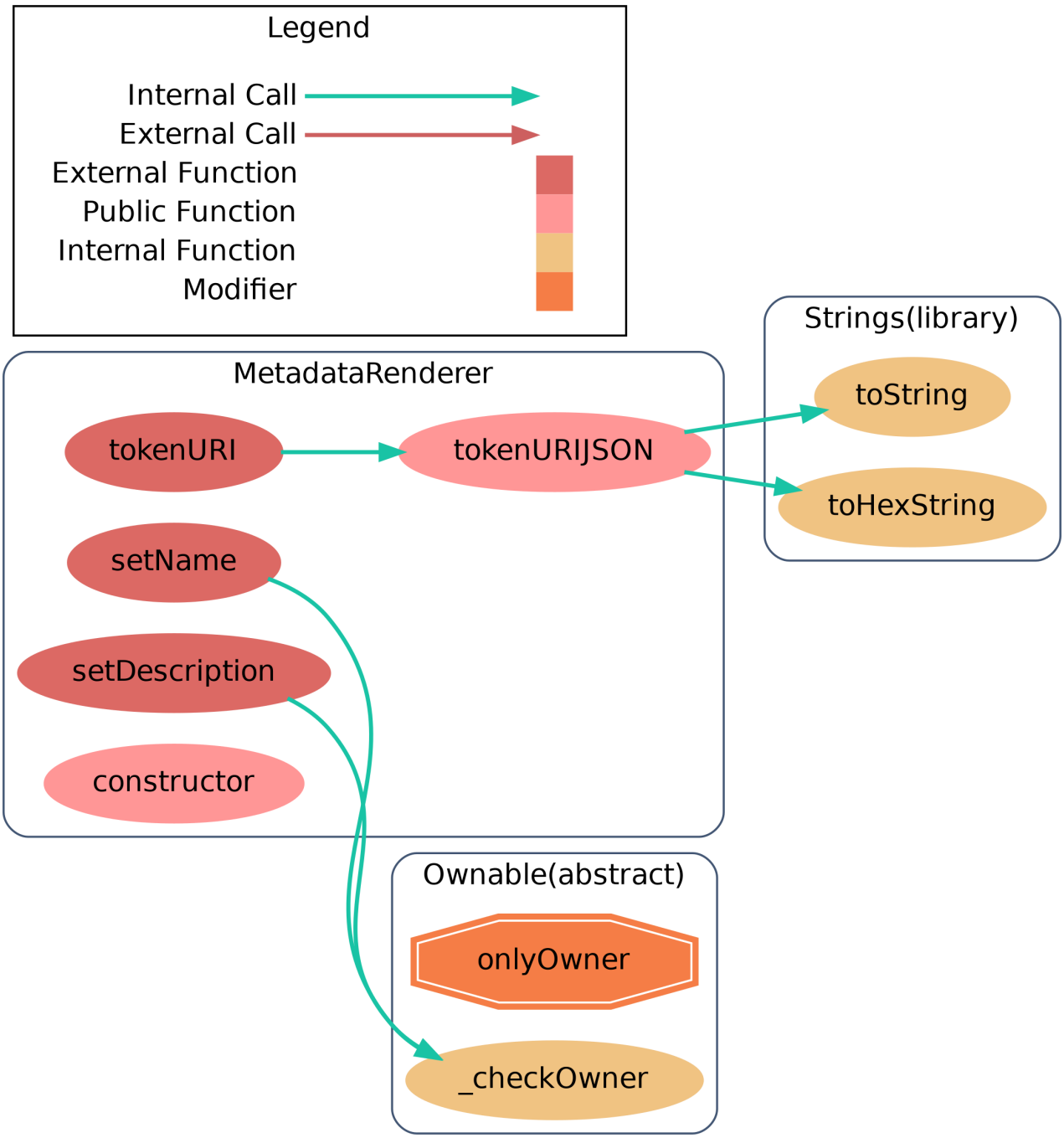
MintRichNFTContract



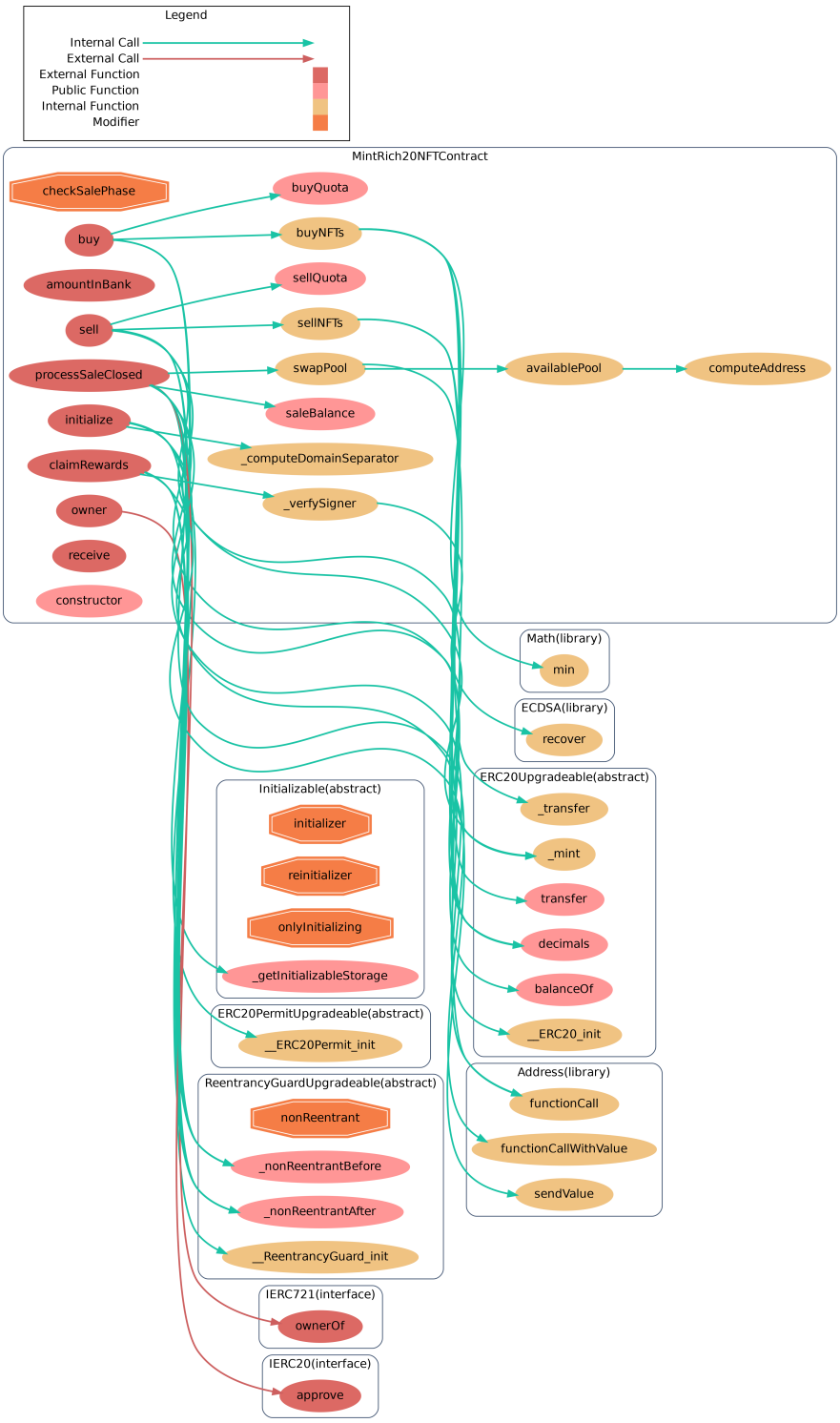
MintRichNFTFactoryContract



MetadataRenderer



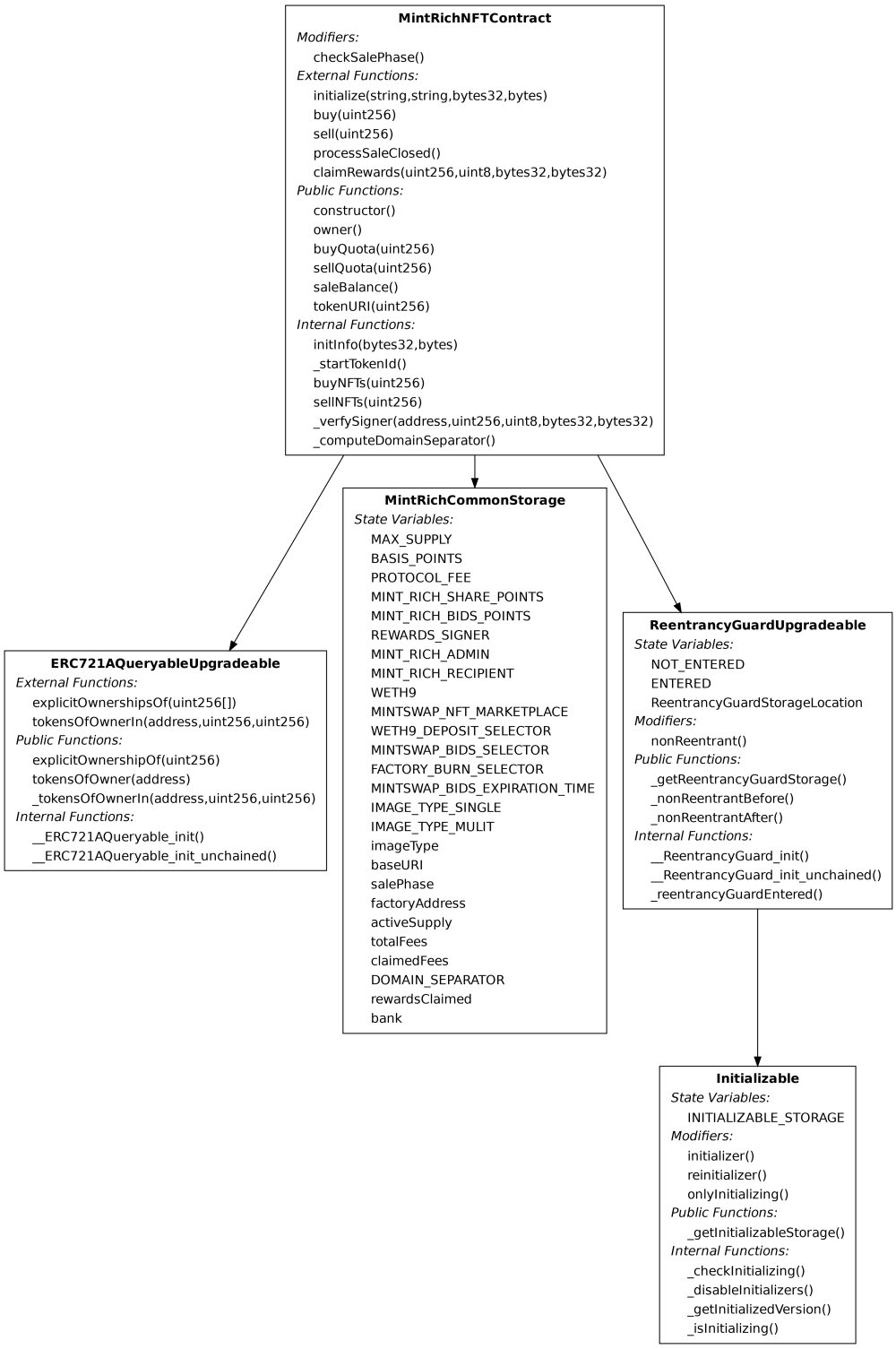
MintRich20NFTContract



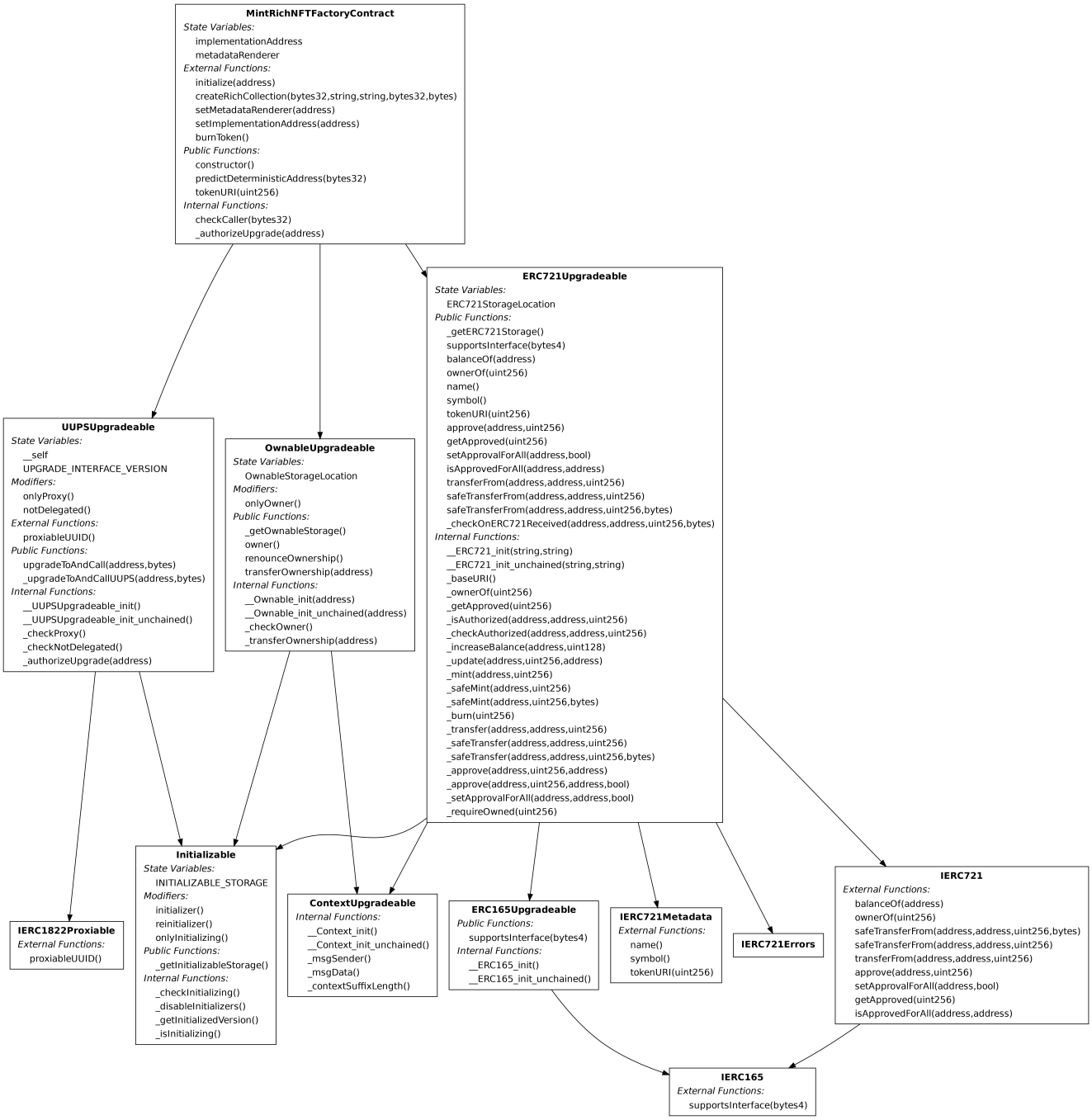
5. Appendix

5.3 Inheritance Graph

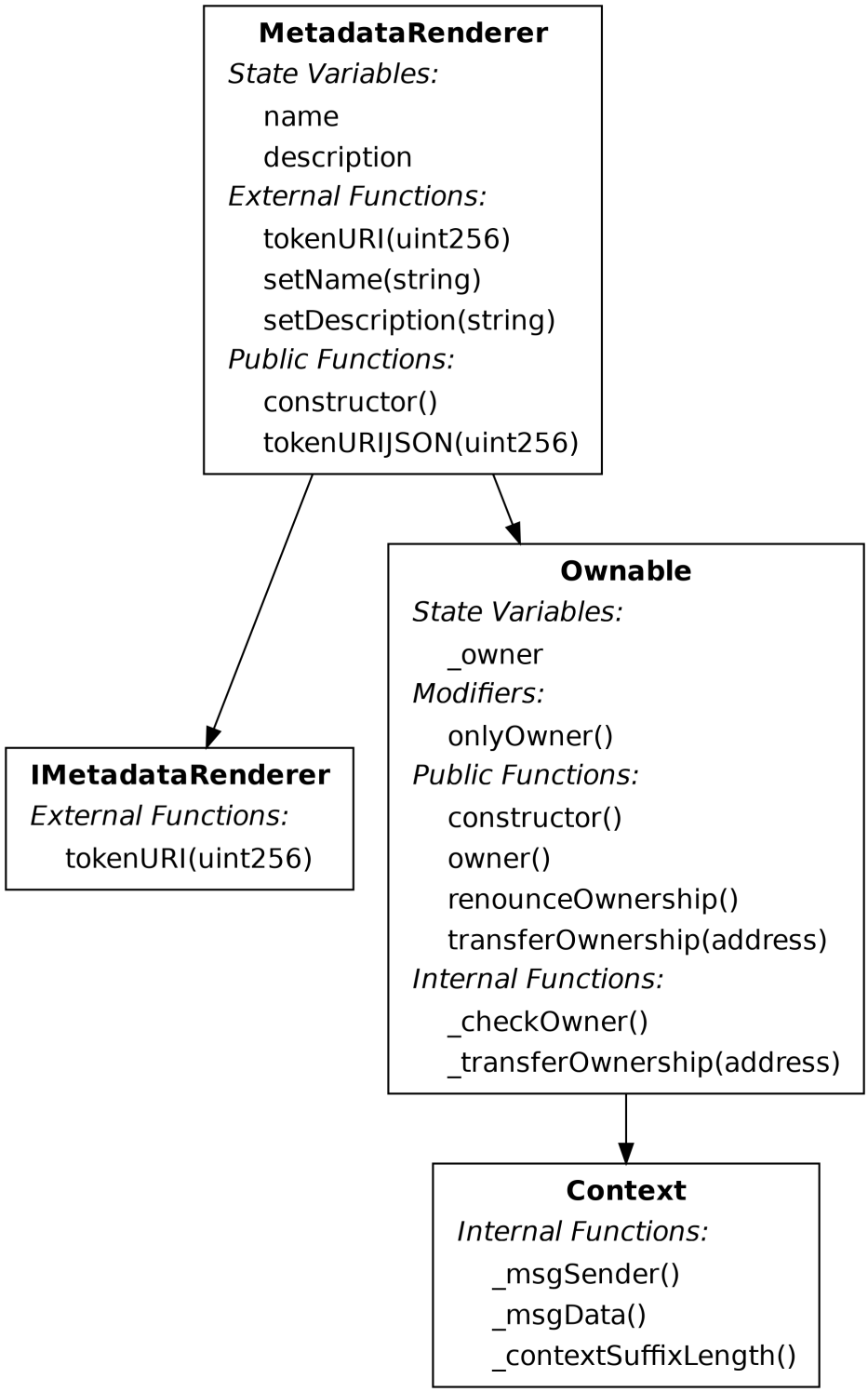
MintRichNFTContract



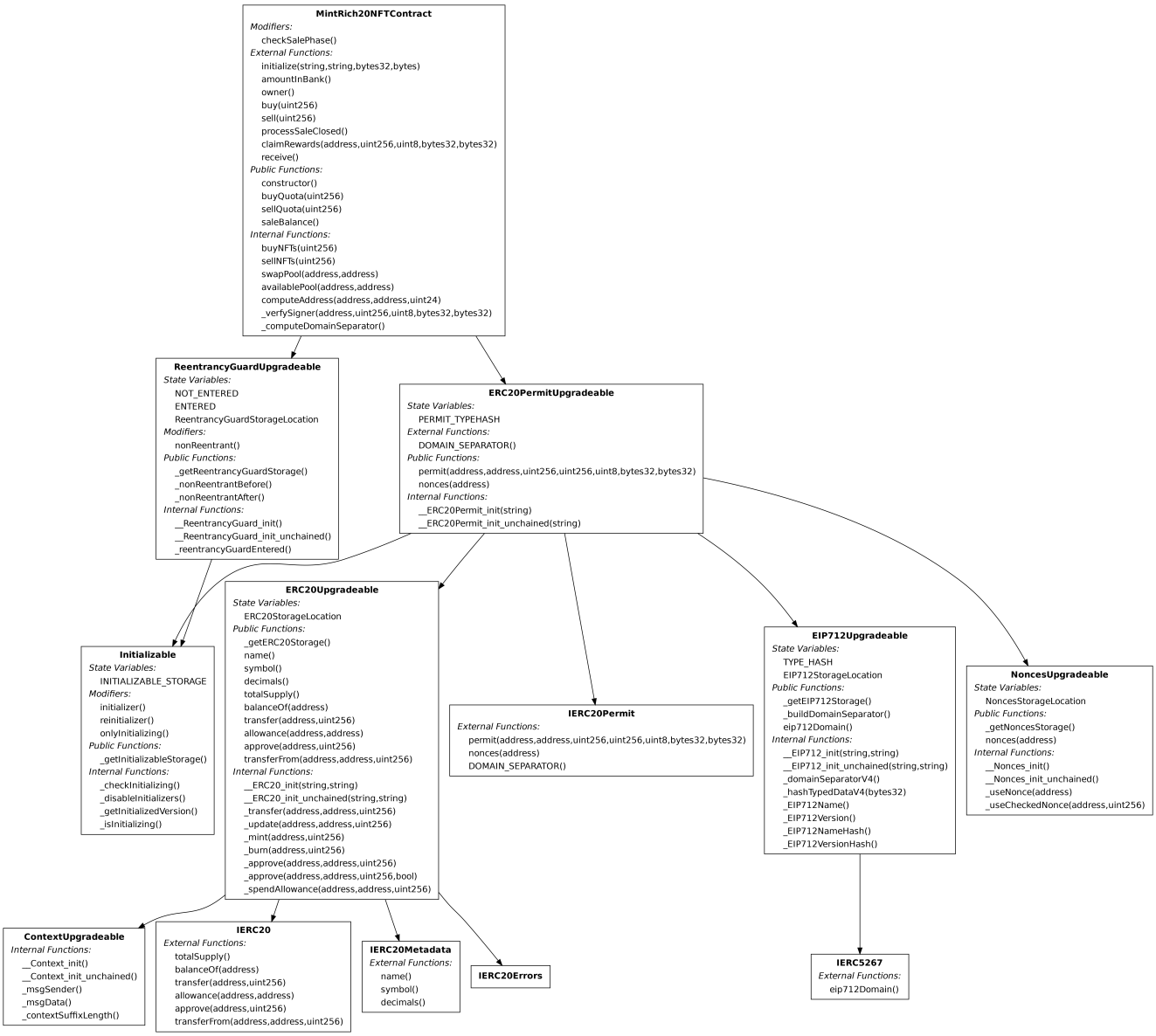
MintRichNFTFactoryContract



MetadataRenderer



MintRich20NFTContract



5.4 Formal Verification Metadata

1. The function `createRichCollection` must succeed only if the caller's address matches the first 20 bytes of the `collectionId`.

```
/// #if_succeeds {:msg "The function `createRichCollection` must succeed
only if the caller's address matches the first 20 bytes of the
`collectionId`."} msg.sender == bytes20(bytes32(uint256(msg.sender))) ==>
checkCaller(collectionId) == true;
function createRichCollection(
    bytes32 collectionId,
    string calldata name,
    string calldata symbol,
    bytes32 packedData,
    bytes calldata information
) external {
```

Passed.

2. Post-creation, the ownership of the newly minted NFT representing the collection must be transferred to the function caller.

```
/// #if_succeeds {:msg "Post-creation, the ownership of the newly minted
NFT representing the collection must be transferred to the function
caller."} ownerOf(uint256(uint160(collection))) == msg.sender;
function createRichCollection(
    bytes32 collectionId,
    string calldata name,
    string calldata symbol,
    bytes32 packedData,
    bytes calldata information
) external {
```

Passed.

3. On a successful `buy` operation, the active supply should increase by the `amount` of NFTs bought.

```
/// #if_succeeds {:msg "On a successful `buy` operation, the active supply
should increase by the `amount` of NFTs bought."} activeSupply ==
old(activeSupply) + amount;
function buy(uint256 amount) external payable nonReentrant checkSalePhase {
```

Passed.

4. When the `buy` function concludes successfully, the `totalFees` collected must encompass the `fees` calculated for the purchase.

```
/// #if_succeeds {:msg "When the `buy` function concludes successfully, the
`totalFees` collected must encompass the `fees` calculated for the
purchase."} totalFees == old(totalFees) + fees;
function buy(uint256 amount) external payable nonReentrant checkSalePhase {
```

Passed.

5. Post a successful `buy`, if there is an overflow of payment (`msg.value > totalPrices`), the surplus is refunded back to the `msg.sender`.

```
/// #if_succeeds {:msg "Post a successful `buy`, if there is an overflow of
payment (`msg.value > totalPrices`), the surplus is refunded back to the
`msg.sender`."} (msg.value > totalPrices) ==> payable(msg.sender).balance
== old(payable(msg.sender).balance) + (msg.value - totalPrices);
function buy(uint256 amount) external payable nonReentrant checkSalePhase {
```

Passed.

6. If the `buy` method completes execution, and if the contract possesses NFTs in its `bank`, these NFTs should be transferred out reducing the `bank`'s length accordingly.

```
/// #if_succeeds {:msg "If the `buy` method completes execution, and if the
contract possesses NFTs in its `bank`, these NFTs should be transferred out
reducing the `bank`'s length accordingly."} bank.length() <
old(bank.length()) && !bank.empty();
function buy(uint256 amount) external payable nonReentrant checkSalePhase {
```

Passed.

7. Following a successful `buy`, the `buyNFTs` internal function should have led to a token transfer, either from the bank or freshly minted, resulting in the `balanceOf(msg.sender)` increasing by `amount`.

```
/// #if_succeeds {:msg "Following a successful `buy`, the `buyNFTs`
internal function should have led to a token transfer, either from the bank
or freshly minted, resulting in the `balanceOf(msg.sender)` increasing by
`amount`."} balanceOf(msg.sender) == old(balanceOf(msg.sender)) + amount;
function buy(uint256 amount) external payable nonReentrant checkSalePhase {
```

Passed.

8. On successful execution of `buyNFTs`, the active supply of NFTs must increase by the `amount` if no tokens are withdrawn from the bank, or by the remaining `mintAmount` after withdrawals.

```
/// #if_succeeds {:msg "On successful execution of `buyNFTs`, the active
supply of NFTs must increase by the `amount` if no tokens are withdrawn
from the bank, or by the remaining `mintAmount` after withdrawals."}
old(activeSupply) + amount == activeSupply && bank.empty();
function buyNFTs(uint256 amount) internal {
```

Passed.

9. When the function `sell` concludes successfully, it necessitates that the active supply is decreased by the sold amount.

```
/// #if_succeeds {:msg "When the function `sell` concludes successfully, it
necessitates that the active supply is decreased by the sold amount."}
old(activeSupply) == activeSupply + amount;
function sell(uint256 amount) external nonReentrant checkSalePhase {
```

Passed.

10. Post a successful `sell` operation, the total fees accumulated by the contract must reflect the addition of fees generated from this transaction.

```
/// #if_succeeds {:msg "Post a successful `sell` operation, the total fees
accumulated by the contract must reflect the addition of fees generated
from this transaction."} old(totalFees) == totalFees - fees;
function sell(uint256 amount) external nonReentrant checkSalePhase {
```

Passed.

11. On executing the `sell` function without errors, the transferred value back to the seller must equal the calculated received prices.

```
/// #if_succeeds {:msg "On executing the `sell` function without errors,
the transferred value back to the seller must equal the calculated received
prices."} payable(msg.sender).balance == old(payable(msg.sender).balance) +
receivedPrices;
function sell(uint256 amount) external nonReentrant checkSalePhase {
```

Passed.

12. Following a successful `sell`, the balance of NFTs owned by the seller must be reduced by the sold amount.

```
/// #if_succeeds {:msg "Following a successful `sell`, the balance of NFTs
owned by the seller must be reduced by the sold amount."}
```

```
balanceOf(msg.sender) == old(balanceOf(msg.sender)) - amount;
function sell(uint256 amount) external nonReentrant checkSalePhase {
```

Passed.

13. On successful execution of the function `sellNFTs`, the active supply of NFTs should be reduced by the `amount` of NFTs being sold.

```
/// #if_succeeds {:msg "On successful execution of the function `sellNFTs`,
the active supply of NFTs should be reduced by the `amount` of NFTs being
sold."} old(activeSupply) == activeSupply + amount;
function sellNFTs(uint256 amount) internal {
```

Passed.

14. In the event of a successful `sellNFTs` operation, the caller's balance of NFTs should decrease by the `amount` specified.

```
/// #if_succeeds {:msg "In the event of a successful `sellNFTs` operation,
the caller's balance of NFTs should decrease by the `amount` specified."}
old(balanceOf[msg.sender]) == balanceOf[msg.sender] + amount;
function sellNFTs(uint256 amount) internal {
```

Passed.

15. For every NFT sold through `sellNFTs`, the ownership should be appropriately transferred from the seller to the contract itself.

```
/// #if_succeeds {:msg "For every NFT sold through `sellNFTs`, the
ownership should be appropriately transferred from the seller to the
contract itself."} forall(uint i in range(amount))
old(ownerOf(tokensOfOwner[msg.sender][i])) == msg.sender &&
ownerOf(tokensOfOwner[msg.sender][i]) == address(this);
function sellNFTs(uint256 amount) internal {
```

Passed.

16. When claiming rewards, the total claimed rewards for the recipient should accurately reflect the update after claiming, and the claimed fees should correspondingly increase by the claimed amount.

```
/// #if_succeeds {:msg "When claiming rewards, the total claimed rewards
for the recipient should accurately reflect the update after claiming, and
the claimed fees should correspondingly increase by the claimed amount."}
rewardsClaimed[recipient] == old(rewardsClaimed[recipient]) + totalRewards
- rewardsClaimed[recipient] && claimedFees == old(claimedFees) +
totalRewards - rewardsClaimed[recipient];
function claimRewards(
```

```
    uint256 totalRewards,
    uint8 _v,
    bytes32 _r,
    bytes32 _s
) external nonReentrant {
```

Passed.

17. The function `claimRewards` should enforce that the claimed amount does not surpass the available unclaimed fees.

```
/// #if_succeeds {:msg "The function `claimRewards` should enforce that the
claimed amount does not surpass the available unclaimed fees."}
totalRewards - rewardsClaimed[recipient] <= old(totalFees) -
old(claimedFees);
function claimRewards(
    uint256 totalRewards,
    uint8 _v,
    bytes32 _r,
    bytes32 _s
) external nonReentrant {
```

Passed.

18. The reward claim should originate from a verified signer as confirmed by the `_verfySigner` function.

```
/// #if_succeeds {:msg "The reward claim should originate from a verified
signer as confirmed by the `_verfySigner` function."}
_verfySigner(recipient, totalRewards, _v, _r, _s) == REWARDS_SIGNER;
function claimRewards(
    uint256 totalRewards,
    uint8 _v,
    bytes32 _r,
    bytes32 _s
) external nonReentrant {
```

Passed.