Immutable Internal Contract Audit

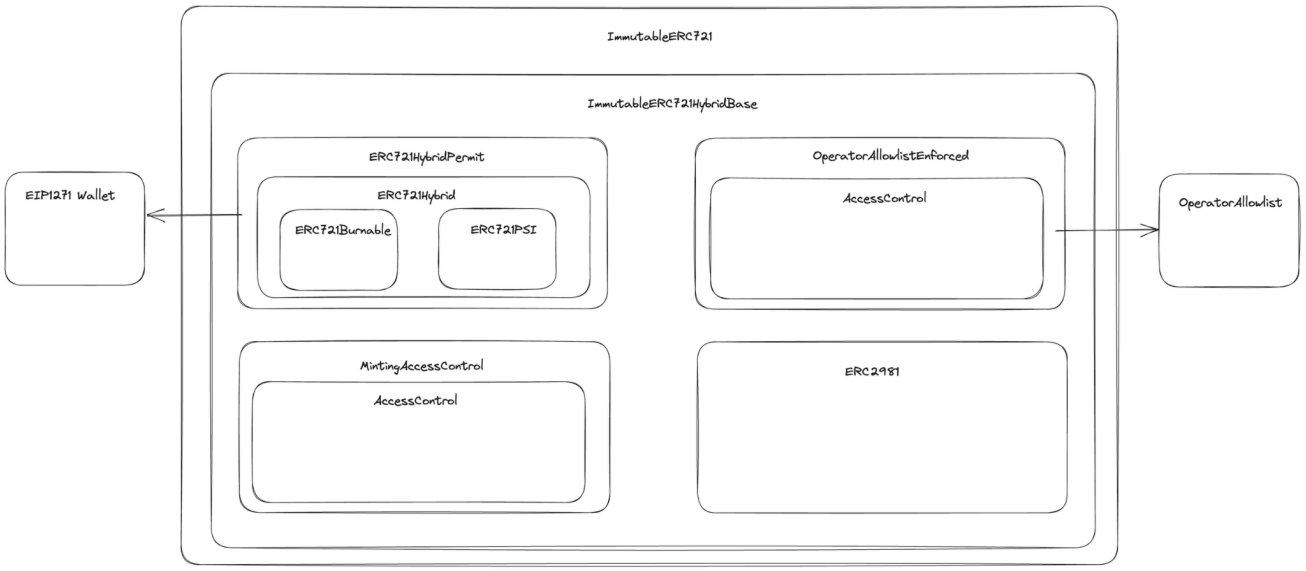
SAR-48: ERC721 Preset Contract

Internal Auditors	Peter Robinson Ryan Teoh
Assessment Date	28 Nov 2023
Initial Report (v13)	December 6, 2023
Final Report (v15)	December 13, 2023.
Revised	Feb 6, 2024

Contents

Description

New ERC721 preset royalty enforcement contracts that allows for batch minting, burning and transferring by id.



Scope

Commit: [8ae72094ab335c6a88ebabde852040e85cb77880]

Asset	Description
Smart Contracts	https://github.com/immutable/contracts/tree/main/contracts/token/erc721
Threat model	https://github.com/immutable/contracts/blob/main/audits/202309-threat-model-preset-erc721.md

Team's Greatest Concerns

Unwanted access to royalty allow list update.

- Unwanted access to burning/transfer/minting tokens
- Unwanted access for admin updates
- Royalty: Compromised royalty allow list allows malicious contracts to transfer assets
- Preset: Compromised ownership allow burning of assets without permissions
- Allowlist: Compromised allow list allows malicious users to add arbitrary addresses into allowed contracts

Basic Contract Analysis

The table below analyzes the contracts.

Type	File	Lines	nLines	nSLOC	Comment Lines	Complexity Score
Contract	contracts/token/erc721/erc721psi/ ERC721Psi.sol	468	462	219	188	209
Abstract	contracts/token/erc721/erc721psi/ ERC721PsiBurnable. sol	84	84	36	38	33
Contract	contracts/token/erc721/x/Asset.sol	22	18	14	1	8
Interface	contracts/token/erc721/x/IMintable.sol	10	5	3	1	3
Library	contracts/token/erc721/x/utils/Bytes.sol	92	84	52	24	54
LIbrary	contracts/t oken/erc721/x/utils/Minting.sol	26	22	15	4	9
Abstract	contracts/t oken/erc721/x/Mintable.sol	42	33	25	1	19
Contract	contracts/token/erc721/preset/ ImmutableERC721.sol	103	103	61	28	49
Contract	contracts/token/erc721/preset/ ImmutableERC721MintByID.sol	80	80	50	18	51
Interface	contracts/token/erc721/abstract/IERC4494.sol	28	18	4	16	9
Abstract	contracts/t oken/erc72 1/abstract/ Immutable ERC721Base.sol	259	239	145	52	131
Abstract	contracts/t oken/erc72 1/abstract/ ERC721HybridPermit. sol	167	167	86	61	58
Abstract	contracts/t oken/erc721/abstract/ Immutable ERC721Hy bridBase.s ol	110	87	58	14	49
Abstract	contracts/token/erc721/abstract/ ERC721Permit.sol	214	188	103	62	60
Abstract	contracts/t oken/erc721/abstract/ ERC721Hy brid.sol	367	351	232	62	190
Abstract	contracts/token/erc72 1/abstract/ MintingAccessControl .sol	30	30	19	5	34
	Totals	**2102**	**1971**	**1122**	**575**	**966**

Smart Contract Interfaces

ERC721Psi

ERC721Psi implements: Context, ERC165, IERC721, IERC721Metadata

Function	Access Modifier	Mutate State
_startTokenId	Internal	
_nextTokenId	Internal	
_totalMinted	Internal	
supportsInterface	Public	
balanceOf	Public	
ownerOf	Public	
_ownerAndBatchHeadOf	Internal	
name	Public	
symbol	Public	
tokenURI	Public	
_baseURI	Internal	
approve	**Public**	**yes**
getApproved	Public	
setApprovalForAll	**Public**	**yes**
isApprovedForAll	Public	
transferFrom	**Public**	**yes**
safeTransferFrom	**Public**	**yes**
safeTransferFrom	**Public**	**yes**
_safeTransfer	Internal	yes
_exists	Internal	
_isApprovedOrOwner	Internal	
_safeMint	Internal	yes
_safeMint	Internal	yes
_mint	Internal	yes
_transfer	Internal	yes

_approve	Internal	yes
_checkOnERC721Received	Private	yes
_getBatchHead	Internal	
totalSupply	Public	
_beforeTokenTransfers	Internal	yes
_afterTokenTransfers	Internal	yes

ERC721PsiBurnable

ERC721PsiBurnable implements ERC721Psi.

Function	Access Modifier	Mutate State
_burn	Internal	yes
_exists	Internal	
totalSupply	Public	
_burned	Internal	
_popcount	Private	

Asset

Asset implements ERC721, Mintable.

Function	Access Modifier	Mutate State
_mintFor	Internal	yes

IMintable

IMintable is an interface.

Function	Access Modifier	Mutate State
mintFor	**External**	**yes**

Bytes

Bytes is a library.

Function	Access Modifier	Mutate State
fromUint	Internal	
indexOf	Internal	
substring	Internal	
toUint	Internal	

Minting

Minting is a library.

Function	Access Modifier	Mutate State
split	Internal	

Mintable

Mintable implements Ownable, IMintable.

Function	Access Modifier	Mutate State
mintFor	**External**	**yes**
_mintFor	Internal	yes

ImmutableERC721

ImmutableERC721 implements ImmutableERC721HybridBase

Function	Access Modifier	Mutate State
mint	**External**	**yes**
safeMint	**External**	**yes**
mintByQuantity	**External**	**yes**
safeMintByQuantity	**External**	**yes**
mintBatchByQuantity	**External**	**yes**

safeMintBatchByQuantity	**External**	**yes**
mintBatch	**External**	**yes**
safeMintBatch	**External**	**yes**
safeBurnBatch	**External**	**yes**
safeTransferFromBatch	**External**	**yes**

IERC4494

IERC4494 is an interface that implements IERC165.

Function	Access Modifier	Mutate State
permit	External	yes
nonces	External	
DOMAIN_SEPARATOR	External	

ImmutableERC721Base

ImmutableERC721Base implements OperatorAllowlistEnforced, ERC721Permit, ERC2981

Function	Access Modifier	Mutate State
_baseURI	Internal	
supportsInterface	Public	
getAdmins	Public	
setBaseURI	**Public**	**yes**
setContractURI	**Public**	**yes**
setApprovalForAll	**Public**	**yes**
_approve	Internal	yes
_transfer	Internal	yes
setDefaultRoyaltyReceiver	**Public**	**yes**
setNFTRoyaltyReceiver	**Public**	**yes**
setNFTRoyaltyReceiverBatch	**Public**	**yes**
grantMinterRole	**Public**	**yes**
revokeMinterRole	**Public**	**yes**

totalSupply	Public	
burn	**Public**	**yes**
safeBurn	**Public**	**yes**
_safeBurnBatch	**Public**	**yes**
_batchMint	Internal	yes
_safeBatchMint	Internal	yes
_mint	Internal	yes
_safeMint	Internal	yes

ERC721HybridPermit

ERC721HybridPermit implements ERC721Hybrid, IERC4494, EIP712.

Function	Access Modifier	Mutate State
permit	**External**	**yes**
_permit	Internal	yes
nonces	External	
DOMAIN_SEPARA TOR	External	
_buildPermitDigest	Internal	
_isValidEOASignat ure	Private	
_isValidERC1271Si gnature	Private	
supportsInterface	Public	
_transfer	Internal	yes

ImmutableERC721HybridBase

ImmutableERC72 1HybridBase implements OperatorAllowlistEnforced, MintingAccessControl, ERC2981, ERC721HybridPermit.

Function	Access Modifier	Mutate State
supportsInterface	Public	
_baseURI	Internal	
setBaseURI	**Public**	**yes**
setContractURI	**Public**	**yes**
setApprovalForAll	**Public**	**yes**
_approve	Internal	yes
_transfer	Internal	yes
setDefaultRoyaltyReceiver	**Public**	**yes**
setNFTRoyaltyRec eiver	**Public**	**yes**
setNFTRoyaltyRec eiverBatch	**Public**	**yes**

ERC721Permit

ERC721Permit implements ERC721Burnable, IERC4494, EIP712, IImmutableERC721Errors.

Function	Access Modifier	Mutate State
permit	**External**	**yes**
_permit	Internal	yes
nonces	External	
DOMAIN_SEPARA TOR	External	
_buildPermitDigest	Internal	
_isValidEOASignat ure	Private	
_isValidERC1271Si gnature	Private	
supportsInterface	Public	
_transfer	Internal	yes

ERC721Hybrid

ERC721Hybrid implements ERC721PsiBurnabl e, ERC721, IImmutableERC721Errors

Function	Access Modifier	Mutate State
mintBatchByQuantityThreshold	**Public**	**yes**
_startTokenId	Internal	
_mintByQuantity	Internal	yes
_safeMintByQuantit y	Internal	yes
_mintBatchByQuan tity	Internal	yes
_safeMintBatchBy Quantity	Internal	yes
_mintByID	Internal	yes
_safeMintByID	Internal	yes
_mintBatchByID	Internal	yes
_safeMintBatchByID	Internal	yes
_mintBatchByIDTo Multiple	Internal	yes
_safeMintBatchByI DToMultiple	Internal	yes
exists	Public	
burn	**Public**	**yes**
burnBatch	External	yes
safeBurn	**Public**	**yes**
_safeBurnBatch	Internal	yes
_exists	Internal	
_transfer	Internal	yes

ownerOf	Public	
_burn	Internal	yes
_approve	Internal	yes
_isApprovedOrOwner	Internal	
_safeTransfer	Internal	yes
safeTransferFrom	**Public**	**yes**
isApprovedForAll	Public	
getApproved	Public	
approve	**Public**	**yes**
transferFrom	**Public**	**yes**
_safeMint	Internal	yes
_safeMint	Internal	yes
_mint	Internal	yes
balanceOf	Public	
totalSupply	Public	
tokenURI	Public	
name	Public	
symbol	Public	
supportsInterface	Public	
_baseURI	Internal	

setApprovalForAll	**Public**	**yes**
safeTransferFrom	**Public**	**yes**

MintingAccessControl

MintingAccessControl implements AccessControlEnumerable.

Function	Access Modifier	Mutate State
getAdmins	Public	
grantMinterRole	**Public**	**yes**
revokeMinterRole	**Public**	**yes**

Abuser Test Cases and Findings

EIP-4494 Implementation

Description: Failure to implement EIP-4494 may lead to other contracts not able to determine if this is EIP 4494 supported.

Comment: The interface is implemented correctly according to the specification.
https://github.com/immutable/contr acts/blob/main/contracts/token/erc 721/abstract/IERC4494.sol

Action Required: None

Status: DONE

Access Control

Description: Are Access Controls being implemented correctly?

Comment: Grants DEFAULT_ADMIN_ROLE to the supplied owner address and has the ability to manage collection, manage MINTER_ROLE and manage OperatorAllowList .
Create a separate role for the allow list management. In this way, DEFAULT_ADMIN only adds and removes roles.
Action Required: Create a separate role to allow list management.

Status: DONE

Token Burn Abuse

Description: Ensure Burn functionality are implemented correctly

Comment: safeBurn uses an extension of Openzeppelin's ERC721 - contracts/token/ERC721/extensions/ERC721Burnable.sol

```
function burn(uint256 tokenId) public virtual {
  // Setting an "auth" arguments enables the `_isAuthorized` check
  // which verifies that the token exists
  // (from != 0). Therefore, it is not needed to verify that the
  // return value is not 0 here.
  _update(address(0), tokenId, _msgSender());
}
```

_update() will check ownership

The _update will check if _msgSender() is the owner or approved to burn the tokenId.

Action Required: Remove unnecessary verification since _update() will check ownership.

Status: DONE

EIP-712 Implementation

Description: Incorrect implementation of EIP 712 may be prone to replay attack.

Comment: Openzeppelin's EIP-712 library was used to implement EIP-712.

Action Required: None.

Status: DONE

Business Requirement

Description:

- Studios should be able to mint multiple tokens efficiently to multiple addresses.
- Studios should be able to mint by token id out of order for metadata association.
- Minting should be restricted to addresses that were granted the minter role.
- Only allowed operators should be able to modify and assign roles to addresses for administering the collection on chain.
- Contracts should not be upgradeable to prevent external developers from getting around royalty requirements.
- Minting should be restricted to addresses that were granted the minter role.

Comment:

- All collection management functionality requires DEFAULT_ADMIN_ROLE
- Contracts should not be upgradeable to prevent external developers from getting around royalty requirements.
- ERC721 preset is not upgradable.
- All Public minting functions either require MINTER_ROLE , IMX or owner() .
- Only allowed operators should be able to modify and assign roles to addresses for administering the collection on chain.

Action Required: None.

Status: DONE

Reentrancy Vulnerabilities

Description: A reentrancy attack is a type of vulnerability in smart contracts that allows attackers to execute a function multiple times before the previous call completes. This can lead to unexpected and harmful behavior, such as the theft of funds or unauthorized access to data.

Comment: Apply the check-effects interactions pattern (http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).

contracts/token/erc721/preset/ImmutableERC721MintByID.sol#33

```
function safeMint(address to, uint256 tokenID) external
onlyRole(MINTER_ROLE) {
   _safeMint(to, tokenID, "");
    _totalSupply++;
}
```

contracts/token/erc721/abstract/ERC721Hybrid.sol#108

```
function _safeMintByID(address to, uint256 tokenId) internal {
        if (tokenId >= mintBatchByQuantityThreshold()) {
            revert IImmutableERC721IDAboveThreshold(tokenId);
    }
    if (_burnedTokens.get(tokenId)) {
            revert IImmutableERC721TokenAlreadyBurned(tokenId);
        }
        ERC721._safeMint(to, tokenId);
        _idMintTotalSupply++;
}
```

contracts/token/erc721/abstract/ImmutableERC721Base.sol#234

```
function _safeBatchMint(IDMint memory mintRequest) internal {
        if (mintRequest.to == address(0)) {
            revert IImmutableERC721SendingToZerothAddress();
        }
        for (uint256 j; j < mintRequest.tokenIds.length; j++) {
            _safeMint(mintRequest.to, mintRequest.tokenIds[j]);
        }
        _totalSupply = _totalSupply + mintRequest.tokenIds.length;
    }
```

Action Required: Apply the check-effects-interactions pattern (Security Considerations — Solidity 0.4.21 documentation).

Status: Done

Solidity Version Requirement

Description: Ensure Solidity the version is used correctly.

Comment: Change the Solidity version to 0.8.19 (and not ^0.8.19). The zkEVM is not compatible with 0.8.20, so using the precise version is important.

Action Required: Change the Solidity version to 0.8.19 (and not ^0.8.19).

Status: Done.

Slither

Description: Run slither on erc721 presets

Comment: No major findings from Slither

Number of lines: 3262 (+ 5362 in dependencies, + 145 in tests)Number of assembly lines: 0
Number of contracts: 31 (+ 40 in dependencies, + 7 tests)

Number of optimization issues: 8
Number of informational issues: 215

Use: Openzeppelin-Ownable, Openzeppelin-ERC20, Openzeppelin-ERC721ERCs: ERC721, ERC20, ERC165

Action Required: None.

Status: Done.

Copyright and License

Description: Ensure correct copyright notice and license is included in each smart contract.

Action Required: Done

Findings Update Feb 6, 2024

ImmutableERC721Base's _safeBurnBatch public

Description: ImmutableERC721Base's _safeBurnBatch function is public. This function should be internal.

Comment: This does not pose a security risk as authentication checks occur during ERC721's _update function.

Action: Deprecate _safeBurnBatch in documentation. Plan to make the function internal in the next major release of software.

Status: In progress.