Immutable Internal Contract Audit

Operator Allowlist

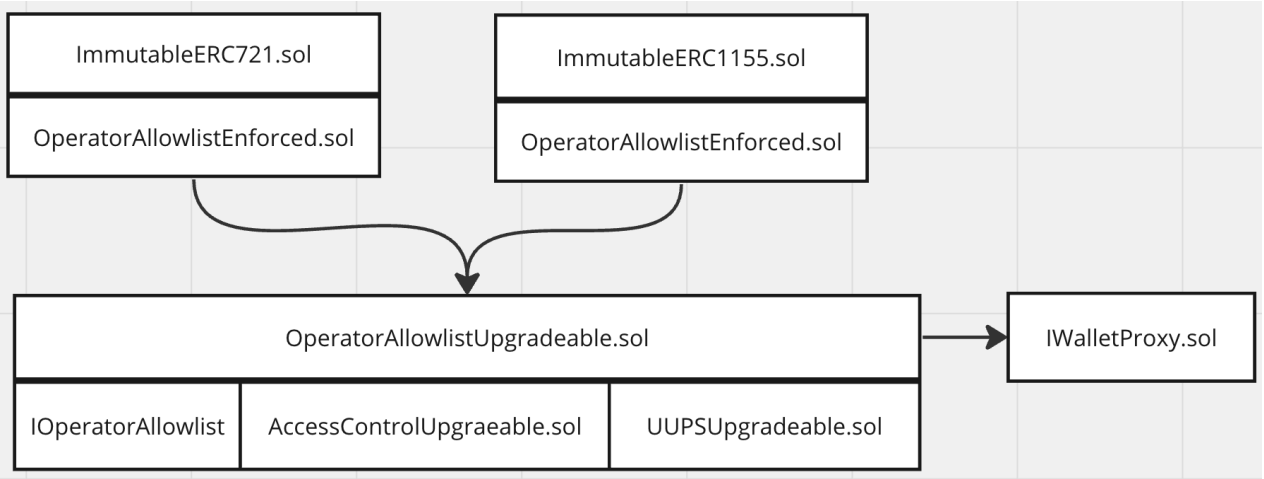
Internal Auditors	Peter Robinson
Assessment Date	February 16, 2024
Final Report	February 16, 2024

Previous audits: The Operator Allowlist code was reviewed twice in 2023 in isolation, plus reviewed in the context of the ERC 721 contracts. This report however is the first write-up of an audit of this code.

Contents

Description

The Operator Allowlist ensures that token contracts are used with approved trading platforms that honor royalty payments. The architecture of the Operator Allowlist system is shown below.



ERC 721 and ERC 1155 contracts extend the OperatorAllowlistEnforced contract. The OperatorAllowlistEnforced contract includes the Solidity modifiers validateApproval and validateTransfer. The token contracts ensure that these modifiers are called whenever an approval or transfer takes place. The modifiers call a function isAllowlisted in the OperatorAllowlistUpgradeable contract.

The OperatorAllowlistUpgradeable contract is a Universal Upgradeable Proxy Standard (UUPS) upgrade contract, and has access control features, and implements the IOperatorAllowlist interface. OperatorAllowlistEnforced calls OperatorAllowlistUpgradeable based on functions defined in the interface.

The OperatorAllowlistUpgradeable's isAllowlisted function checks to see if the address has been approved, or if the contract at the address is a wallet proxy contract that is using a valid wallet implementation. It does this by checking that the code hash has been approved, and then calling the function PROXY_getImplementation on the proxy to check if the implementation address has been approved.

Scope

Commit: [c3dc85984ced2e8e75576ba5ef84f2141d810bb4]

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

Team’s Greatest Concerns

- Allowlist is bypassed.
- Allowlist is disabled.

Basic Contract Analysis

The table below analyzes[1] the contracts.

Type	File	Lines	nSLOC	Complexity
Contract	OperatorAllowlistUpgradeable.sol	180	78	86
Abstract	OperatorAllowlistEnforced.sol	112	42	32
Interface	IOperatorAllowlist.sol	13	3	3

[1] Analysis produced using the Solidity Metrics VSCode extension.

Smart Contracts

OperatorAllowlistUpgradeable

OperatorAllowlistUpgradeable extends: ERC165, AccessControlEnumerableUpgradeable, UUPSUpgradeable, IOperatorAllowlist.

Roles:

Name	Purpose
DEFAULT_ADMIN_ROLE	Administer other roles.
UPGRADE_ROLE	Authorize the upgrade of the contract.
REGISTRAR_ROLE	Add and remove addresses and bytecode hashes from the allow lists.

External or Public Functions[2] that modify state:

Function Name	Function Selector	Authorisation Check	Notes
initialize	c0c53b8b	initializer modifier	
addAddressesToAllowlist	7f17caa7	REGISTRAR_ROLE	
addWalletToAllowlist	0488d974	REGISTRAR_ROLE	
grantRole	2f2ff15d	Role admin / DEFAULT_ADMIN_ROLE	
removeAddressesFromAllowlist	6caacbe7	REGISTRAR_ROLE	
removeWalletFromAllowlist	6aa70096	REGISTRAR_ROLE	
renounceRole	36568abe	msg.sender	Only msg.sender can renounce their own role.
revokeRole	d547741f	Role admin / DEFAULT_ADMIN_ROLE	
upgradeTo	3659cfe6	UPGRADE_ROLE	External function calls _authorizeUpgrade which does the auth check
upgradeToAndCall	4f1ef286	UPGRADE_ROLE	External function calls _authorizeUpgrade which does the auth check

[2] The list of functions was determined using forge inspect ./<path>/<contract>.sol:<contract> methods

External or Public Functions[3] that do not modify state:

Function Name	Function Selector	Notes
DEFAULT_ADMIN_ROLE	a217fddf	Inherited from AccessControlEnumerableUpgradeable
REGISTRAR_ROLE	f68e9553	Could be private.
UPGRADE_ROLE	b908afa8	Could be private.
getRoleAdmin	248a9ca3	
getRoleMember	9010d07c	
getRoleMemberCount	ca15c873	
hasRole	91d14854	
isAllowlisted	05a3b809	
proxiableUUID	52d1902d	
supportsInterface	01ffc9a7	

Upgradeable checks: All inherited contracts are designed for upgrade. Contract includes a storage gap at the end of the contract.

Analysis of code logic: Appears to be correct.

OperatorAllowlistEnforced

OperatorAllowlistEnforced is an abstract contract that extends: AccessControlEnumerable, OperatorAllowlistEnforcementErrors.

Roles:

Name	Purpose
DEFAULT_ADMIN_ROLE	Administer other roles.

External or Public Functions[4] that modify state:

Function Name	Function Selector	Authorisation Check	Notes

[3] The list of functions was determined using forge inspect ./<path>/<contract>.sol:<contract> methods
[4] The list of functions was determined using forge inspect ./<path>/<contract>.sol:<contract> methods

initialize	c0c53b8b	initializer modifier	
grantRole	2f2ff15d	Role admin / DEFAULT_ADMIN_ROLE	
renounceRole	36568abe	msg.sender	Only msg.sender can renounce their own role.
revokeRole	d547741f	Role admin / DEFAULT_ADMIN_ROLE	

External or Public Functions[5] that do not modify state:

Function Name	Function Selector	Notes
DEFAULT_ADMIN_ROLE	a217fddf	Inherited from AccessControlEnumerable
getRoleAdmin	248a9ca3	
getRoleMember	9010d07c	
getRoleMemberCount	ca15c873	
hasRole	91d14854	
operatorAllowlist	29326f29	
supportsInterface	01ffc9a7	

Upgradeable checks: Not upgradeable.

Analysis of code logic: Appears to be correct.

[5] The list of functions was determined using forge inspect ./<path>/<contract>.sol:<contract> methods

Abuser Test Cases and Findings

Improve efficiency of OperatorAllowlistEnforced

Classification: Gas Efficiency

Description: OperatorAllowlistEnforced's validatorApproval and validateTransfer call isAllowlisted on the OperatorAllowlistUpgradeable contract between zero and three times, depending on the result of various checks. It would be more gas efficient to determine which addresses needed to be checked and then do one function call, rather than potentially do multiple calls.

Action: Review the gas improvement of this the next time the code is being upgraded and consider implementing the upgrade.

Status: In review by team.

OperatorAllowlistEnforced should not extend AccessControlEnumerable

Classification: Informational

Description: OperatorAllowlistEnforced no longer needs to extend AccessControlEnumerable as it no longer uses the features of this contract.

Action: Remove the importing and extending of AccessControlEnumerable from OperatorAllowlistEnforced the next time the code is being upgraded.

Status: In review by team.

Unneeded Public Constants

Classification: Gas Efficiency

Description: UPGRADE_ROLE and REGISTRAR_ROLE could be private. Doing this would mean that there are two fewer function selectors. Solidity compiled bytecode starts by doing multiple if-else blocks checking each possible function selector to determine which function to jump to. Having additional unneeded functions public results in more checks, which wastes a small amount of gas.

Action: Make UPGRADE_ROLE and REGISTRAR_ROLE private.

Status: In review by team.

Non-Allowed Contracts Receive Tokens

Classification: Informational

Description: The following scenario would allow a contract that is not in the allow list to be the recipient of a token.

- An EOA or Passport Wallet transfers an ERC 721 from themselves to an address which is an empty account. As it is an empty account, to.code.length returns 0, so the transfer is approved by the OperatorAllowlistEnforced.
- The account's address is actually a CFA for a contract which we wouldn't put on the allow list.
- When the deployer allow list restriction is removed (no date currently set for this), the contract is deployed to the address.
- At this point, a non-approved contract now owns an ERC 721.

This scenario, though interesting, is not deemed an issue. The non-approved contract could not then transfer the token or approve the token.

Action: None.