Cyberscope

Audit Report

O3 Swap

March 2023

Wrapper.sol	43a22a6fd60883aa68cf00f7b0ed6315bfccf5aba2ed6eb054191a51de6096e8
LockProxy.sol	ae6ef6c251307856cdc121d02dfb57864cb569aa23d31152e34d2708a0aec82e
Pool.sol	75a12fec6673cfd50012e79667024d74e23285dd082d3904ce072e30b487e0de
PNFT.sol	262d9ac69b6385f9c0f35cc068afd6f1c2170adeda3d271dcac42288593b72dc
ClaimUtil.sol	1d7c0ea463c77dd3a9b81566760c01f23f91200d04636eeb9f9a5e4babf8f489

Audited by © cyberscope

Table of Contents

Review

Audit Updates

Initial Audit	06 Mar 2023

Source Files

Filename	SHA256
access/Ownable.sol	6b16de7f11e6ed1ef831e77046a52f4f84024be258c7d9d89452643dd41befed
claim/ClaimUtil.sol	1d7c0ea463c77dd3a9b81566760c01f23f91200d04636eeb9f9a5e4babf8f489
interfaces/IEthCrossChainManager.sol	ea7bfe8601ad1d00ae50df205ab5d746a68aaefa5631bee1fc5ff669f2d946bd
interfaces/IEthCrossChainManagerProxy.sol	b7530d39fe73a46f4073a31939e2d947a22d3b9189a99eeb6b041978f533c48a
interfaces/ILockProxy.sol	a75b6f1c96f7a8470b8318af74aad5ca1d84ea81f7a82a7f3c40b9e963e789f6
interfaces/IPNFT.sol	44d4929fac00ae0d064bcec7b24aa8e72e94623319b4f40c846eebb8cd8d2177
interfaces/IPool.sol	084d422837940362bd637b9f720d5f5c06482d1108a9ff304c5db9e745571c9b
LockProxy.sol	ae6ef6c251307856cdc121d02dfb57864cb569aa23d31152e34d2708a0aec82e
PNFT.sol	262d9ac69b6385f9c0f35cc068afd6f1c2170adeda3d271dcac42288593b72dc
Pool.sol	75a12fec6673cfd50012e79667024d74e23285dd082d3904ce072e30b487e0de
Utils.sol	5a05511b03b28037d002bf038cffe2ad470c6e8ff70ecaffb08060b2a4937842
Wrapper.sol	43a22a6fd60883aa68cf00f7b0ed6315bfccf5aba2ed6eb054191a51de6096e8

Introduction

The `O3 Swap` ecosystem consists of various contracts and implements a cross-chain NFT transfer functionality. For the scope of this audit, we focused on the following contracts:

- ClaimUtil.sol
- LockProxy.sol
- PNFT.sol
- Pool.sol
- Wrapper.sol

Each one of those contracts has its role in cross-chain asset transfer functionality.

ClaimUtil Contract

The purpose of this contract is to provide a utility for handling NFTs in a more modular and safe way. The `fetchAndReturn` modifier allows a function to fetch NFTs from a specific pool, execute a function, and then return the NFTs back to the pool, reducing the risk of losing NFTs.

Roles

Owner

The owner has authority over the following functions:

- `function rescueNFT(address nft, uint tokenId)`

LockProxy Contract

The `LockProxy` contract provides functionality for locking and unlocking NFTs (non-fungible tokens) across different blockchains. The contract has several functions for managing pools, binding assets, and proxy hashes.

Roles

Owner

The owner has authority over the following functions:

- `function setManagerProxy(address _managerProxy)`
- `function setPool(address _poolAddress)`
- `function setPoolBatch(address[] calldata _poolAddresses)`
- `function unsetPool(address nftAddress)`
- `function bindProxyHash(uint64 toChainId, bytes calldata targetProxyHash)`
- `function bindAssetHash(address fromAssetHash, uint64 toChainId, bytes calldata toAssetHash)`
- `function bindProxyHashBatch(uint64[] calldata toChainIds, bytes[] calldata targetProxyHashes)`
- `function bindAssetHashBatch(address[] calldata fromAssetHashs, uint64[] calldata toChainIds, bytes[] calldata toAssetHashes)`

ManagerContract

The manager contract has authority over the following functions:

- `function unlock(bytes calldata argsBs, bytes calldata fromContractAddr, uint64 fromChainId)`

PNFT Contract

The `PNFT` contract is a non-fungible token (NFT) contract that allows minting NFTs with a URI and transferring them between accounts. The contract also allows for setting a lock proxy address that is allowed to mint new NFTs. `PNFT` does not support burning of tokens, and it is owned by the contract creator who can set the token URI and the lock proxy address.

Roles

Owner

The owner has authority over the following functions:

- `function setLockProxy(address _lockProxy)`
- `function setTokenURI(uint256 tokenId, string calldata _tokenURI)`

LockProxy

The lock proxy has authority over the following functions:

- `function mintWithURI(address to, uint256 tokenId, string calldata uri)`

Pool Contract

The `Pool` contract defines a pool for storing and managing ownership of non-fungible tokens (NFTs).

Roles

Owner

The owner has authority over the following functions:

- `function setPooledNFT(address _nftAddress)`
- `function setLockProxy(address _lockProxy)`
- `function setAuthorizedCaller(address _caller)`
- `function unsetAuthorizedCaller(address _caller)`
- `function rescueNFT(address nft, uint tokenId)`
- `function claimAirdropETH()`
- `function claimAirdropToken(address token)`
- `function claimAirdropNFT(address asset, uint tokenId)`
- `function claimAirdropNFTs(address asset)`
- `function claimAirdropNFTs(address asset, uint[] calldata tokenIds)`

LockProxy

The lock proxy has authority over the following functions:

- `function store(address nftOwner, uint tokenId)`
- `function withdraw(uint tokenId, address toAddress)`

AuthCallers

The auth callers have authority over the following functions:

- `function externalCall(address callee, bytes calldata callData)`

- `function setApproveForAirdrop(address nftOwner)`

Wrapper Contract

The `Wrapper` contract is designed to be used in a multi-chain ecosystem where ERC721 tokens need to be transferred between different blockchains. This can happen when a user wants to move their assets from one blockchain to another or when a decentralized application needs to access assets from multiple blockchains.

Roles

Owner

The owner has authority over the following functions:

- `function setFeeCollector(address _feeCollector)`
- `function setLockProxy(address _lockProxy)`
- `function pause()`
- `function unpause()`
- `function rescueFund(address tokenAddress)`

FeeCollector

The fee collector has authority over the following functions:

- `function extractFee()`

User

The user can interact with the following functions:

- `function lock(address asset, uint64 toChainId, address toAddress, uint256 tokenId)`

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	CR	Code Repetition	Unresolved
●	LUV	Lack Of Unique Verification	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L19	Stable Compiler Version	Unresolved

CR - Code Repetition

Criticality	Minor / Informative
Location	LockProxy.sol#L53,63 Pool.sol#L44
Status	Unresolved

Description

The contract contains repetitive code segments. There are potential issues that can arise when using code segments in Solidity. Some of them can lead to issues like gas efficiency, complexity, readability, security, and maintainability of the source code. It is generally a good idea to try to minimize code repetition where possible.

The `LockProxy` contract can set a new `_poolAddress` by calling the `setPool()` function. Since the `Pool` contract contains the `lockProxy` address, it could notify the `LockProxy` contract of this change instead of repeating the same functionality in both contracts.

```
function _setPool(address _poolAddress) internal {
    require(_poolAddress != address(0), "pool address cannot be zero");
    require(IPool(_poolAddress).lockProxy() == address(this), "lockProxy address
configured in pool does not match");

    address nftAddress = IPool(_poolAddress).pooledNFT();
    require(nftAddress != address(0), "pooledNFT configured in pool cannot be
zero");

    pools[nftAddress] = _poolAddress;
    emit SetPoolEvent(nftAddress, _poolAddress);
}
```

Recommendation

The team is advised to avoid repeating the same code in multiple places, which can make the contract easier to read and maintain. The authors could try to reuse code wherever possible, as this can help reduce the complexity and size of the contract. For instance, the contract could reuse the common code segments in an internal function in order to avoid repeating the same code in multiple places.

LUV - Lack Of Unique Verification

Criticality	Minor / Informative
Location	LockProxy.sol#L116,145
Status	Unresolved

Description

The contract functions `onERC721Received` and `unlock` do not have any mechanism to verify the authenticity of the bytes calldata parameters. This can potentially lead to replay attacks. A replay attack occurs when an attacker captures a valid message and sends it again to the same contract to perform a transaction without the sender's consent. This attack can be prevented by adding a nonce to the transaction and storing its value in the contract, which ensures that each transaction is unique and can only be executed once.

```
function onERC721Received(address, address from, uint256 tokenId, bytes calldata
data) public nonReentrant override returns (bytes4) { ... }
...
function unlock(bytes calldata argsBs, bytes calldata fromContractAddr, uint64
fromChainId) external onlyManagerContract nonReentrant returns (bool) { ... }
```

Recommendation

The team is advised to include a unique identifier like a nonce in the contract and verify its authenticity before executing any transaction.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Wrapper.sol#L38,45 Pool.sol#L44,52,59,64 PNFT.sol#L25,38 LockProxy.sol#L46,53,57
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address _feeCollector
address _lockProxy
address _nftAddress
address _caller
string calldata _tokenURI
address _managerProxy
address _poolAddress
address[] calldata _poolAddresses
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	PNFT.sol#L51
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function _burn(uint256 tokenId) internal override(ERC721, ERC721URIStorage) {}
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	LockProxy.sol#L210
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
TxArgs memory args
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Wrapper.sol#L3 Pool.sol#L3 PNFT.sol#L4 LockProxy.sol#L3 interfaces/IPool.sol#L3 interfaces/IPNFT.sol#L3 interfaces/ILockProxy.sol#L3
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
ClaimUtil	Implementation	Ownable, IERC721Receiver, ReentrancyGuard		
	rescueNFT	External	✓	onlyOwner nonReentrant
	fetchNFTs	Private	✓	nonReentrant
	returnNFTs	Private	✓	nonReentrant
	onERC721Received	Public	✓	-
IEthCrossChainManager	Interface			
	crossChain	External	✓	-
IEthCrossChainManagerProxy	Interface			
	getEthCrossChainManager	External		-
ILockProxy	Interface			
	managerProxyContract	External		-

	getCrossChainManagerAddress	External		-
	proxyHashMap	External		-
	assetHashMap	External		-
	pools	External		-
	setManagerProxy	External	✓	-
	setPool	External	✓	-
	unsetPool	External	✓	-
	bindProxyHash	External	✓	-
	bindAssetHash	External	✓	-
IPNFT	Interface			
	lockProxy	External		-
	tokenURI	External		-
	setLockProxy	External	✓	-
	mintWithURI	External	✓	-
	setTokenURI	External	✓	-
IPool	Interface			
	pooledNFT	External		-
	lockProxy	External		-
	authorizedCallers	External		-
	store	External	✓	-
	withdraw	External	✓	-
	ownerOf	External		-
	balanceOf	External		-
	tokenOfOwnerByIndex	External		-
	externalCall	External	✓	-
	setApproveForAirdrop	External	✓	-
	claimAirdropETH	External	✓	-

	claimAirdropToken	External	✓	-
	claimAirdropNFT	External	✓	-
	claimAirdropNFTs	External	✓	-
	claimAirdropNFTs	External	✓	-
LockProxy	Implementation	IERC721Receiver, Ownable, ILockProxy, ReentrancyGuard		
	setManagerProxy	External	✓	onlyOwner
	setPool	External	✓	onlyOwner
	setPoolBatch	External	✓	onlyOwner
	_setPool	Internal	✓	
	unsetPool	External	✓	onlyOwner
	bindProxyHash	External	✓	onlyOwner
	bindAssetHash	External	✓	onlyOwner
	bindProxyHashBatch	External	✓	onlyOwner
	bindAssetHashBatch	External	✓	onlyOwner
	unlock	External	✓	onlyManagerContract nonReentrant
	onERC721Received	Public	✓	nonReentrant
	_onERC721Received	Internal	✓	
	getCrossChainManagerAddress	Public		-
	_serializeTxArgs	Internal		
	_deserializeTxArgs	Internal		
	_deserializeCallData	Internal		
IUSDC	Interface			
	mint	External	✓	-

ERC20Airdrop	Implementation			
	claim	External	✓	-
	reset	External	✓	-
IERC20Airdrop	Interface			
	claim	External	✓	-
ClaimERC20Airdrop	Implementation	ClaimUtil		
	claimAirdrop	External	✓	fetchAndReturn
EnaAirdrop	Implementation			
	claim	External	✓	-
IEnaAirdrop	Interface			
	claim	External	✓	-
ClaimNFTAirdrop	Implementation	ClaimUtil		
	claimAirdrop	External	✓	fetchAndReturn
	onERC721Received	Public	✓	-
ERC20Token	Implementation	ERC20		
		Public	✓	ERC20
	decimals	Public		-
	setAuthorizedCaller	External	✓	-
	mint	External	✓	onlyCallers
IHackerPunk	Interface	IERC721, IERC721Enumerable		

	getO3Burned	External		-
	mint	External	✓	-
	setBaseURI	External	✓	-
	setTokenURI	External	✓	-
	isMintCallerAuthorized	External		-
	setAuthorizedMintCaller	External	✓	-
	removeAuthorizedMintCaller	External	✓	-
	pause	External	✓	-
	unpause	External	✓	-
	withdraw	External	✓	-
	totalSupply	External		-
BoredApeYachtClub	Implementation	ERC721URIStorage, ERC721Enumerable, Ownable		
		Public	✓	ERC721
	withdraw	Public	✓	onlyOwner
	reserveApes	Public	✓	onlyOwner
	setRevealTimestamp	Public	✓	onlyOwner
	setProvenanceHash	Public	✓	onlyOwner
	setBaseURI	External	✓	onlyOwner
	_baseURI	Internal		
	setTokenURI	External	✓	onlyOwner
	tokenURI	Public		-
	flipSaleState	Public	✓	onlyOwner
	mintApe	Public	Payable	-
	setStartingIndex	Public	✓	-
	emergencySetStartingIndexBlock	Public	✓	onlyOwner
	supportsInterface	Public		-

	_burn	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
USDCoin	Implementation	ERC20		
		Public	✓	ERC20
	decimals	Public		-
	setAuthorizedCaller	External	✓	-
	mint	External	✓	onlyCallers
PNFT	Implementation	ERC721URIStorage, ERC721Enumerable, Ownable, IPNFT		
		Public	✓	ERC721
	setLockProxy	External	✓	onlyOwner
	mintWithURI	External	✓	onlyLockProxy
	setTokenURI	External	✓	onlyOwner
	tokenURI	Public		-
	supportsInterface	Public		-
	_burn	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
Pool	Implementation	Ownable, IERC721Receiver, IPool, ReentrancyGuard		
	setPooledNFT	External	✓	onlyOwner
	setLockProxy	External	✓	onlyOwner
	setAuthorizedCaller	External	✓	onlyOwner
	unsetAuthorizedCaller	External	✓	onlyOwner

	rescueNFT	External	✓	nonReentrant onlyOwner
	store	External	✓	nonReentrant onlyLockProxy
	withdraw	External	✓	nonReentrant onlyLockProxy
	ownerOf	External		-
	balanceOf	Public		-
	tokenOfOwnerByIndex	Public		-
	externalCall	External	✓	onlyAuthCallers nonReentrant
	setApproveForAirdrop	External	✓	onlyAuthCallers nonReentrant
	claimAirdropETH	External	✓	onlyOwner
	claimAirdropToken	External	✓	onlyOwner nonReentrant
	claimAirdropNFT	External	✓	onlyOwner nonReentrant
	claimAirdropNFTs	External	✓	onlyOwner nonReentrant
	claimAirdropNFTs	External	✓	onlyOwner nonReentrant
	_claimAirdropNFTs	Internal	✓	
	onERC721Received	Public		-
	_addTokenToOwnerEnumeration	Private	✓	
	_removeTokenFromOwnerEnumeration	Private	✓	
Utils	Library			
	WriteByte	Internal		
	WriteUint8	Internal		
	WriteUint16	Internal		
	WriteUint32	Internal		
	WriteUint64	Internal		
	WriteUint255	Internal		

	WriteVarBytes	Internal		
	WriteVarUint	Internal		
	NextByte	Internal		
	NextUint8	Internal		
	NextUint16	Internal		
	NextUint32	Internal		
	NextUint64	Internal		
	NextUint255	Internal		
	NextVarBytes	Internal		
	NextVarUint	Internal		
	bytesToAddress	Internal		
	equalStorage	Internal		
Wrapper	Implementation	Ownable, Pausable, Reentrancy Guard		
	setFeeCollector	External	✓	onlyOwner
	setLockProxy	External	✓	onlyOwner
	pause	External	✓	onlyOwner
	unpause	External	✓	onlyOwner
	extractFee	External	✓	onlyFeeCollector
	rescueFund	External	✓	nonReentrant onlyOwner
	lock	External	Payable	nonReentrant whenNotPaused
	_serializeCallData	Internal		

Inheritance Graph

Flow Graph

Summary

O3swap contract implements an nft, utility, and bridge mechanism. This audit investigates security issues, business logic concerns, and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io