Cyberscope

Audit Report

NFT Shuttle by O3 Labs

March 2023

Wrapper.sol	9b23ba19257d3f437affcc232d3edcc60e8ed69d6ff529e060c51e8feb3b4864
LockProxy.sol	b0c534975863def7bc68f3f072f41eb131accb4f404a6e763fc938c5c96c53cf
Pool.sol	5de869653f2658ec12c44ba1f3d20032efdc48dce11e820c66a937c8acf867f3
PNFT.sol	f932b1006c86d0848591a46d014de5f3952a1a82a68a5eafde11db10c0dac6b0
Claimable.sol	9ef965e72de062b07a9fe7117f69e68fb5291f4f820e777ef8fa8aecc4f48881
Audited by	© cyberscope

Table of Contents

Review

Repository	https://github.com/NFTShuttle/nft-shuttle-contracts
Commit	85a04037119463f4568aee4311261794120e3abe

Audit Updates

Initial Audit	06 Mar 2023
Corrected Phase 2	15 Mar 2023

Source Files

Filename	SHA256
access/Ownable.sol	9e36338aeede7728427a10ad7b3118b4c677e16c13f9a3ca3fb138bacf877bb6
claim/Claimable.sol	9ef965e72de062b07a9fe7117f69e68fb5291f4f820e777ef8fa8aecc4f48881
interfaces/IEthCrossChainManager.sol	cac9f817e1ccf80937f8bf6f5ddf7a4475c142cb160020ac79188546cde1f81f
interfaces/IEthCrossChainManagerProxy.sol	d79305ec024e62197c24e538b4030918d814ef50fdb17296585712356ffbac4b
interfaces/ILockProxy.sol	668354a263e3a40c6c765625a1b73f31886732e26b8522944c08df69acdf523b
interfaces/IPNFT.sol	9478c6abce1573b114f03499a731abf51a0244aa40253582af430e5b4b5b08de
interfaces/IPool.sol	c0522195bd3ef967f97f9bb9ac83a8663e631ba40f5e8f7a9379f20f904df249
LockProxy.sol	b0c534975863def7bc68f3f072f41eb131accb4f404a6e763fc938c5c96c53cf
mock/ERC20Token.sol	02c1c9a8956cbad21fd0587a7e3bce3cbfcef147dc6780f258a0be46861fbcd4
mock/MockClaimERC20Airdrop.sol	d71e59057338f47f8a0f5e093023dfa042720e433d214e2b1dc0f94ab309779a
mock/MockClaimNFTAirdrop.sol	cc88a2ddaffdafecaa7af4e05aabf1c035e80266b062eb8d0cbdb7f2d6fa1099
mock/nfts/BoredApeYachtClub.sol	1d8f1a2400c0499268b773483536e70fbbc10127b751d6d2fefb37f345d4cb9f
mock/StdERC721.sol	60d720fef50700e1347d73b0d51f688c207e5ea7b3fbfb60db317d6fd7dcfc66

PNFT.sol	f932b1006c86d0848591a46d014de5f39 52a1a82a68a5eafde11db10c0dac6b0
Pool.sol	5de869653f2658ec12c44ba1f3d20032ef dc48dce11e820c66a937c8acf867f3
Utils.sol	7e7bd291811b2dfc93425584b69a14b49 565aab576f55ec40b7eb4d4127db6ff
Wrapper.sol	9b23ba19257d3f437affcc232d3edcc60e 8ed69d6ff529e060c51e8feb3b4864

Introduction

The `O3swap` ecosystem consists of various contracts. For the scope of this audit, we focused on the following:

- Claimable.sol
- LockProxy.sol
- PNFT.sol
- Pool.sol
- Wrapper.sol

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

Claimable Contract

The purpose of this contract is to provide a utility for handling NFTs in a more modular and safe way.

Roles

Owner

The owner has authority over the following functions:

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

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 newManagerProxy)`
- `function setPool(address newPool)`
- `function setPoolBatch(address[] calldata newPools)`
- `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)`
- `function pauseUnlock()`
- `function unpauseUnlock()`
- `function pauseReceive()`
- `function unpauseReceive()`

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 NFTs, 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 newLockProxy)`
- `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 newNFT)`
- `function setLockProxy(address newLockProxy)`
- `function setAuthorizedCaller(address caller)`
- `function unsetAuthorizedCaller(address caller)`
- `function setExtCaller(address caller)`
- `function unsetExtCaller(address caller)`
- `function setAllowedExtCallee(address caller)`
- `function unsetAllowedExtCallee(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 setApproveForAirdrop(address nftOwner)`

ExtAuthCallers

The auth callers have authority over the following functions:

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

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 newFeeCollector)`
- `function setLockProxy(address newLockProxy)`
- `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
●	LUV	Lack Of Unique Verification	Acknowledged
●	CR	Code Repetition	Acknowledged
●	L09	Dead Code Elimination	Acknowledged

LUV - Lack Of Unique Verification

Criticality	Minor / Informative
Location	LockProxy.sol#L137,168
Status	Acknowledged

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 nonce in the contract and verify its authenticity before executing any transaction.

Team Update

The team responded with the following statement:

“After consideration, we decide to keep current version. ‘onERC721Received’ is designed to be public to receive NFTs, its inner logic have a complete logic to verify msg.sender, store given NFT to corresponding pool and verify new owner of the NFT, together with the whitelist mechanism, a replay message with invalid msg.sender will revert.”

CR - Code Repetition

Criticality	Minor / Informative
Location	LockProxy.sol#L56,66 Pool.sol#L58
Status	Acknowledged

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.

Team Update

The team responded with the following statement:

“After consideration, we decide to keep current version. After a new pool was deployed, setLockProxy will be executed before register this pool to lockproxy, if there is a notify mechanism, such as invoke a external function from pool to lockproxy, then there must be some kind of validations to verify this contract call is secure, this will increase complexlty.”

L09 - Dead Code Elimination

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

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.

Team Update

The team responded with the following statement:

“PNFT inherits from standard openzeppelin ERC721URIStorage contract, it must override the base function ‘_burn’, also we leave the implementation empty to explicitly tell this PNFT is not burnable.”

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Claimable	Implementation	Ownable, IERC721Receiver, ReentrancyGuard		
		Public	✓	-
	rescueNFT	External	✓	onlyOwner nonReentrant
	setPool	External	✓	onlyOwner
	claimAirdrop	External	✓	nonReentrant noNFTsLeft
	getCaller	Internal		
	isClaiming	Internal		
	_beforeClaim	Internal	✓	
	_execClaim	Internal	✓	
	_afterClaim	Internal	✓	
	fetchNFTs	Private	✓	
	returnNFTs	Private	✓	
	onERC721Received	Public	✓	-
ILockProxy	Interface			
	managerProxyContract	External		-
	getCrossChainManagerAddress	External		-
	proxyHashMap	External		-
	assetHashMap	External		-
	pools	External		-
	setManagerProxy	External	✓	-

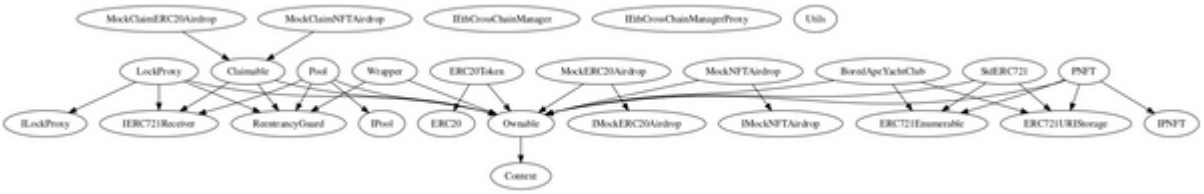
	setPool	External	✓	-
	unsetPool	External	✓	-
	bindProxyHash	External	✓	-
	bindAssetHash	External	✓	-
IPNFT	Interface			
	lockProxy	External		-
	tokenURI	External		-
	setLockProxy	External	✓	-
	mintWithURI	External	✓	-
	setTokenURI	External	✓	-
IPool	Interface			
	pooledNFT	External		-
	lockProxy	External		-
	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	Private	✓	
	unsetPool	External	✓	onlyOwner
	bindProxyHash	External	✓	onlyOwner
	bindAssetHash	External	✓	onlyOwner
	bindProxyHashBatch	External	✓	onlyOwner
	bindAssetHashBatch	External	✓	onlyOwner
	pauseUnlock	External	✓	onlyOwner
	unpauseUnlock	External	✓	onlyOwner
	pauseReceive	External	✓	onlyOwner
	unpauseReceive	External	✓	onlyOwner
	unlock	External	✓	onlyManagerContract nonReentrant
	onERC721Received	Public	✓	nonReentrant
	_onERC721Received	Private	✓	
	getCrossChainManagerAddress	Public		-
	_serializeTxArgs	Internal		
	_deserializeTxArgs	Internal		
	_deserializeCallData	Internal		
PNFT	Implementation	ERC721URIStorage, ERC721Enumerable, Ownable, IPNFT		

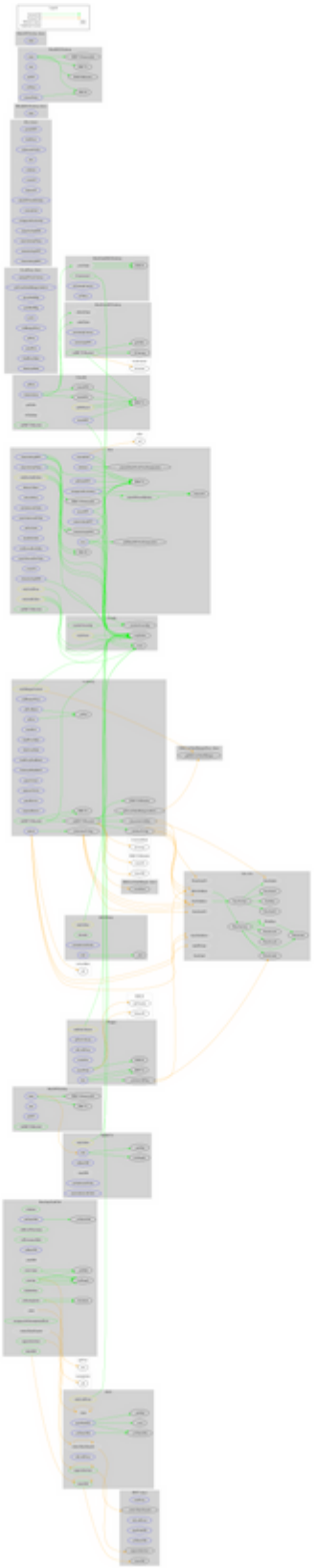
		Public	✓	ERC721
	setLockProxy	External	✓	onlyOwner
	mintWithURI	External	✓	onlyLockProxy
	setTokenURI	External	✓	onlyOwner
	tokenURI	Public		-
	supportsInterface	Public		-
	_burn	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
Pool	Implementation	Ownable, IERC721Receiver, IPool, ReentrancyGuard		
		External	Payable	-
	setPooledNFT	External	✓	onlyOwner
	setLockProxy	External	✓	onlyOwner
	setAuthorizedCaller	External	✓	onlyOwner
	unsetAuthorizedCaller	External	✓	onlyOwner
	setExtCaller	External	✓	onlyOwner
	unsetExtCaller	External	✓	onlyOwner
	setAllowedExtCallee	External	✓	onlyOwner
	unsetAllowedExtCallee	External	✓	onlyOwner
	rescueNFT	External	✓	nonReentrant onlyOwner
	store	External	✓	nonReentrant onlyLockProxy
	withdraw	External	✓	nonReentrant onlyLockProxy
	ownerOf	External		-
	balanceOf	Public		-
	tokenOfOwnerByIndex	Public		-

	externalCall	External	✓	onlyExtAuthCallers 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	✓	
Wrapper	Implementation	Ownable, ReentrancyGuard		
	setFeeCollector	External	✓	onlyOwner
	setLockProxy	External	✓	onlyOwner
	extractFee	External	✓	onlyFeeCollector
	rescueFund	External	✓	nonReentrant onlyOwner
	lock	External	Payable	nonReentrant
	_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