PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero ONFT721

10 Aug 2024

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team. Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for LayerZero ONFT721 on the Ethereum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	LayerZero ONFT721
URL	https://www.layerzero.foundation/
Platform	Ethereum
Language	Solidity
Preliminary	https://github.com/LayerZero-Labs/devtools/tree/3df40c887ea836f77fe008cacf0f460835141ebb/packages/onft-evm-/contracts

1.2 Contracts Assessed

Name	Contract	Live Code Match
ONFT721		PENDING
ONFT721Adapter		PENDING
ONFT721Core		PENDING
ONFT721MsgCodec		PENDING
ONFTComposeMsgCode-c		PENDING

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
Governance	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	1	-	-	1
Informational	5	5	-	-
Total	**6**	**5**	**-**	**1**

Classification of Issues

Severity	Description
Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 ONFT721

ID	Severity	Summary	Status
1	LOW	NFTs from extensions which revert certain transactions such as ones from blocked wallets or soulbound tokens may still be bridgeable	ACKNOWLEDGED
2	INFO	Typographical issues	✓ RESOLVED

1.3.2 ONFT721Adapter

No issues found.

1.3.3 ONFT721Core

ID	Severity	Summary	Status
3	INFO	_lzReceive does not adhere to checks-effects-interactions	✓ RESOLVED
4	INFO	Gas optimizations	✓ RESOLVED

1.3.4 ONFT721MsgCodec

ID	Severity	Summary	Status
5	INFO	Gas optimizations	✓ RESOLVED

1.3.5 ONFTComposeMsgCodec

ID	Severity	Summary	Status
6	INFO	Typographical issues	✓ RESOLVED

2 Findings

2.1 ONFT721

The `ONFT721` represents the main extension of `ONFT721Core`. It represents an NFT which can be natively bridged over the LayerZero network. As described within the `ONFT721Core` portion of this audit, it is the `owner` who can fully configure this nft and link it to ONFT deployments on other chains.

When an ONFT is bridged from one chain with a `ONFT721` deployment to another, it is burned on the source chain and minted on the destination chain. This is the core difference compared to the `ONFT721Adapter`, which instead stores the NFT within its balance. It should be noted that the `ONFT721` contract is the actual NFT itself, while the `ONFT721Adapter` links to an existing NFT.

To send an NFT, the caller needs to be the owner of it. Authorization is insufficient.

2.1.1 Privileged Functions

- `setBaseURI`
- `setMsgInspector`
- `setPreCrime`
- `setEnforcedOptions`
- `setPeer`
- `setDelegate`
- `transferOwnership`
- `renounceOwnership`

2.1.2 Issues & Recommendations

Issue #1	NFTs from extensions which revert certain transactions such as ones from blocked wallets or soulbound tokens may still be bridgeable
Severity	LOW SEVERITY
Description	Similar to ERC20s, certain token implementations may extend the token logic with things such as pausing, banning and soulbound controls (non-transferability of certain NFTs). For example, here are some users discussing how to make certain NFTs soulbound: https://forum.openzeppelin.com/t/soulbound-nft-migration-from-v4-9-to--v5-0/38931/4 By using `_burn`, less metadata of who is initiating this transaction is sent to `_update`, potentially causing such checks to still pass because the auth metadata does not include the `msg.sender`.
Recommendation	Intuitively, re-implementing a `_burn` function which provides the metadata (auth) is desired. However, this is a rough issue to resolve, as re-implementing `_burn` has the downside that users who override `_burn` with additional logic will not have that logic automatically be executed with the LayerZero burn implementation. It may be easiest to simply document this.
Resolution	ACKNOWLEDGED The client has documented this behavior.

Issue #2	Typographical issues
Severity	INFORMATIONAL
Description	Line 37 `function setBaseURI(string memory _baseTokenURI) external onlyOwner {` This function lacks an event. It's argument can furthermore be marked as `calldata`.
Recommendation	Consider fixing the typographical issues.
Resolution	RESOLVED

2.2 ONFT721Adapter

The `ONFT721Adapter` allows for depositing an existing NFT into its balance and then bridging it to `ONFT721` deployments on other chains. These NFTs are stored in the reserves of the adapter until they eventually get bridged back to this source chain.

It is required to only have a single adapter per ONFT deployment, as otherwise the adapter may not have the right nft's in its balance.

As described within the `ONFT721Core` portion of this audit, it is the `owner` who can fully configure this nft and link it to ONFT deployments on other chains.

To send an NFT, the caller needs to be the owner of it. Authorization is insufficient.

2.2.1 Privileged Functions

- `setMsgInspector`
- `setPreCrime`
- `setEnforcedOptions`
- `setPeer`
- `setDelegate`
- `transferOwnership`
- `renounceOwnership`

2.2.2 Issues & Recommendations

No issues found.

2.3 ONFT721Core

The `ONFT721Core` contract represents the main and shared logic between the `ONFT721` and `ONFT721Adapter` contracts. The `owner` of this contract can fully configure the peer deployment contracts on other chains.

To bridge an NFT, users can call the `send` function which allows them to specify the nft token id, the destination chain and the destination address. Furthermore, they can specify a compose message to be executed on the destination address to integrate an action when the NFT is received.

It should be noted that validation on the `to` address format is limited, meaning transactions may be stuck if users miss-specify it. Since execution in LayerZero v2 is unordered, this is not a big deal.

2.3.1 Privileged Functions

- `setMsgInspector`
- `setPreCrime`
- `setEnforcedOptions`
- `setPeer`
- `setDelegate`
- `transferOwnership`
- `renounceOwnership`

2.3.2 Issues & Recommendations

Issue #3	_lzReceive does not adhere to checks-effects-interactions
Severity	INFORMATIONAL

Description	Lines 129-136
	
	`_credit(toAddress, tokenId, _origin.srcEid);`
	
	`if (_message.isComposed()) {`
	`  bytes memory composeMsg = ONFTComposeMsgCodec.encode(-`
	`_origin.nonce, _origin.srcEid, _message.composeMsg());`
	`  // @dev As batching is not implemented, the compose index is always 0.`
	`  // @dev If batching is added, the index will need to be tracked.`
	`  endpoint.sendCompose(toAddress, _guid, 0 /* the index of composed message*/, composeMsg);`
	`}`
	
	The `EndpointV2` implementation does not execute anything, and does not cause any external reentrancy interactions. However, due to `_credit` potentially causing a reentrancy hook being called on the underlying NFT, a malicious user could potentially reenter at that point.
	
	The function therefore does not adhere to checks-effects-interactions.
	
	That being said, inverting the order may have a downside, if reentrancy was permitted during the `_credit`, this means that users could potentially execute the compose message before the funds were credited, which may be even worse.

Recommendation	Consider whether it makes sense to re-order these operations to adhere to the checks-effects-interactions pattern. However, as described above, this is a trade-off. This issue will be resolved even if no changes were made, as we understand the benefit of keeping the code as-is as well.
Resolution	RESOLVED The client has indicated they prefer to keep the code as-is, for the benefits mentioned above. They have furthermore indicated they understand the trade-off of doing this.

Issue #4	Gas optimizations
Severity	INFORMATIONAL
Description	Line 105 `if (msgInspector != address(0)) IOAppMsgInspector(msgInspector).inspect(message, options);` `msgInspector` should be cached as it is fetched from storage twice here, resulting in wasted gas.
Recommendation	Consider fixing the gas optimizations.
Resolution	RESOLVED

2.4 ONFT721MsgCodec

The `ONFT721MsgCodec` is used to encode and decode ONFT bridge transactions into and from LayerZero messages.

2.4.1 Privileged Functions

None.

2.4.2 Issues & Recommendations

Issue #5	Gas optimizations
Severity	INFORMATIONAL
Description	Line 47 `return abi.decode(_msg[SEND_TO_OFFSET:TOKEN_ID_OFFSET], (uint256));` Using `abi.decode` is not only inconsistent with the rest of the codec and other codec's, it also costs slightly more gas compared to doing `uint256(bytes32(…))`.
Recommendation	Consider fixing the gas optimizations.
Resolution	RESOLVED

2.5 ONFTComposeMsgCodec

The `ONFTComposeMsgCodec` is used to encode and decode the compose portion of ONFT LayerZero messages.

2.5.1 Privileged Functions

None.

2.5.2 Issues & Recommendations

Issue #6	Typographical issues
Severity	INFORMATIONAL

Description	Lines 31 and 40
	`* @dev Retrieves the nonce from the composed message.`
	`* @dev Retrieves the source LayerZero endpoint ID from the composed message.`
	These values are not retrieved from the composed message. Instead, they are retrieved from the first portion of the full message, before the compose message portion.

Recommendation	Consider fixing the typographical issues.
Resolution	RESOLVED