# Land and Tunnel Audit

OPENZEPPELIN SECURITY | AUGUST 9, 2023

Security Audits

August 9, 2023

This security assessment was prepared by **OpenZeppelin**.

## Table of Contents

NFT

Timeline
- From 2023-04-03
- To 2023-05-08

Languages
- Solidity

Total Issues
- 34 (25 resolved, 3 partially resolved)

Critical Severity Issues
- 0 (0 resolved)

High Severity Issues
- 1 (1 resolved)

Medium Severity Issues
- 2 (2 resolved)

Low Severity Issues
- 9 (7 resolved, 1 partially resolved)

Notes & Additional Information
- 22 (15 resolved, 2 partially resolved)

## Scope

We audited the thesandboxgame/sandbox-smart-contracts repository at the 8430ea5fdb0f4905b9678689ce0cbc2f74a704b6 commit.

```
src/
├── solc_0.5
│   ├── contracts_common
│   │   ├── BaseWithStorage
│   │   │   ├── MetaTransactionReceiverV2.sol
│   │   │   └── SuperOperatorsV2.sol
│   │   ├── Interfaces
│   │   │   ├── ERC721Events.sol
│   │   │   ├── ERC721MandatoryTokenReceiver.sol
│   │   │   └── ERC721TokenReceiver.sol
│   │   └── Libraries
│   │       └── AddressUtils.sol
│   ├── Land
│   │   └── erc721
│   │       ├── ERC721BaseTokenV2.sol
```

```
│       ├── contracts
│       │   └── upgradeable
│       │       └── OperatorFiltererUpgradeable.sol
│       └── interfaces
│           └── IOperatorFilterRegistry.sol
└── solc_0.8
    ├── common
    │   ├── BaseWithStorage
    │   │   ├── ERC2771Handler.sol
    │   │   ├── ERC721BaseTokenV2.sol
    │   │   ├── WithAdminV2.sol
    │   │   └── WithSuperOperatorsV2.sol
    │   ├── fx-portal
    │   │   ├── FxBaseChildTunnelUpgradeable.sol
    │   │   └── FxBaseRootTunnelUpgradeable.sol
    │   └── interfaces
    │       ├── IERC721MandatoryTokenReceiver.sol
    │       ├── ILandToken.sol
    │       ├── ILandTokenV2.sol
    │       ├── IPolygonLand.sol
    │       ├── IPolygonLandV2.sol
    │       ├── IPolygonLandTunnel.sol
    │       └── IPolygonLandWithSetApproval.sol
    ├── OperatorFilterer
    │   ├── contracts
    │   │   └── upgradeable
    │   │       └── OperatorFiltererUpgradeable.sol
    │   └── interfaces
    │       └── IOperatorFilterRegistry.sol
    └── polygon
        ├── child
        │   └── land
        │       ├── PolygonLandBaseTokenV2.sol
        │       ├── PolygonLandTunnelMigration.sol
        │       ├── PolygonLandTunnelV2.sol
        │       └── PolygonLandV2.sol
```

```
└── LandTunnelV2.sol
```

## System Overview

The LAND token follows the EIP-721 non-fungible token (NFT) standard and has been deployed on both the Ethereum and Polygon networks. LAND represents a part of the 408x408 grid of LAND tokens available as part of The Sandbox game. The grid size is fixed and thus the total LAND tokens available is fixed at 166,464. Each LAND token is identifiable by its coordinates within the grid.

LAND tokens can be combined into quads that represent adjoined 3x3, 6x6, 12x12, or 24x24 groups of LAND tokens. The deployed contracts contain specialized functionality for efficiently transferring and minting quads. A quad can be created by becoming the owner of adjoining LAND tokens and there are no restrictions on separating a quad into smaller child quads, joining it with other quads to form a larger parent quad, or independently transferring individual LAND tokens from within the quad.

Both the Ethereum and Polygon deployments of the LAND token support meta transactions for all publicly exposed functionality including approvals and transferring LAND tokens and quads.

The Polygon implementation has been deployed with a proxy pattern and is thus upgradeable by the proxy admin. The Ethereum version is not upgradeable.

This audit contains a thorough review of both the Ethereum and Polygon LAND token contracts, as well as their dependencies. The main update besides some minor enhancements to the contracts was the integration of the OpenSea Operator Filter Registry. This registry prevents what contracts can have approvals set for them, as well as what contracts can call `transferFrom` on behalf of a LAND token holder. This is intended to penalize exchanges that do not pay royalties on sales of the tokens although the LAND contract does not implement EIP-2891 for the NFT royalty standard.

In addition to owning the LAND tokens on each of the Ethereum and Polygon networks, The Sandbox allows users to transfer their quads between these networks. The LAND TUNNEL suite of contracts integrates with the Polygon Fx-Portal bridge to allow the bilateral movement of tokens.

On the Ethereum network, the user calls the `batchTransferQuadToL2` function to transfer a batch of their quad to the LAND Tunnel contract. This function in turn sends a message to the Polygon network where equivalent quads are minted and transferred to the user's provided address.

A similar process is followed on the Polygon network where the user calls the `batchTransferQuadToL1` function to transfer a batch of quads to the Polygon-LAND Tunnel contract, which signals the Ethereum network to mint and transfer equivalent quads to the user's provided address. While the transfer of quads from Ethereum to Polygon is unbounded by the system, there exists a limit on the gas spent and on the size of quads that can be transferred in a single transaction.

The Tunnel Migration contract exists on both networks to allow an admin to transfer the LAND tokens and quads from an old Tunnel contract to a new implementation, on the respective networks. Additionally, the Tunnel Migration contract on the Polygon network allows the admin to transfer the LANDs locked in the old Tunnel to a new Tunnel contract and further transfer them to the Ethereum network in the same function call.

To ensure the security and reliability of the system, audits have been conducted on the TUNNEL-related and migration contracts.

## Privileged Roles

The audited contracts contain many privileged roles that are required for properly managing the LAND token and associated tunnel contracts.

The `LandV3` contract contains the following privileged roles:

- The `admin` role has the following capabilities:
  - Can set the `_metaTransactionContracts` with the `setMetaTransactionProcessor` function.
  - Can update the operator filterer registry using the `register` and `setOperatorRegistry` functions.
  - Can add and remove `minters` using the `setMinter` function.

- The `_superOperators` role has the following capabilities:
  - Can set approval of a LAND token held by any account for any `operator` using the `approveFor` and `approve` functions.
  - Can set approval for all of an account's LAND tokens for any account using the `setApprovalForAllFor` function.
  - Can transfer LAND tokens from any account to any account using the `transferFrom`, `safeTransferFrom`, `batchTransferQuad`, `transferQuad`, `batchTransferFrom`, and `safeBatchTransferFrom` functions.
  - Can burn the LAND tokens held by any account using the `burnFrom` function.
- The `_minters` role has the following capabilities:
  - Can mint new LAND tokens with the `mintQuad` and `mintAndTransferQuad` functions.

The `PolygonLandV2` contract contains the following privileged roles:

- The `admin` role has the following capabilities:
  - Can set the `_trustedForwarder` with the `setTrustedForwarder` function.
  - Can update the operator filterer registry using the `register` and `setOperatorRegistry` functions.
  - Can add and remove `minters` using the `setMinter` function.
  - Can add and remove `_superOperators` using the `setSuperOperator` function.
  - Can change the admin with the `changeAdmin` function.
- The `_superOperators` role has the following capabilities:
  - Can set approval of a LAND token held by any account for any `operator` using the `approveFor` and `approve` functions.
  - Can set approval for all of an account's LAND tokens for any account using the `setApprovalForAllFor` function.
  - Can transfer LAND tokens from any account to any account using the `transferFrom`, `safeTransferFrom`, `batchTransferQuad`,

- The `_minters` role has the following capabilities:
  - Can mint new LAND tokens with the `mintQuad` and `mintAndTransferQuad` functions.

The `LandTunnelV2` contract contains the following privileged roles:

- The `owner` role has the following capabilities:
  - Can change ownership with the `transferOwnership` function.
  - Can renounce ownership with the `renounceOwnership` function.
  - Can set the `_trustedForwarder` with the `setTrustedForwarder` function.
  - Can pause and unpause the contract using the `pause` and `unpause` functions which prevent new tokens from being bridged to Polygon.

The `PolygonLandTunnelV2` contract contains the following privileged roles:

- The `owner` role has the following capabilities:
  - Can change ownership with the `transferOwnership` function.
  - Can renounce ownership with the `renounceOwnership` function.
  - Can set the gas limits for transactions with the `setMaxLimitOnL1`, `setLimit`, and `setupLimits`
  - Can set the maximum number of LAND tokens that can be bridged in a single transaction using the `setMaxAllowedQuads` function.
  - Can set the `_trustedForwarder` with the `setTrustedForwarder` function
  - Can pause and unpause the contract using the `pause` and `unpause` functions which prevent new tokens from being bridged to Ethereum.

The `LandTunnelMigration` contract contains the following privileged roles:

- The `admin` role has the following capabilities:
  - Can change the admin of the contract with the `changeAdmin` function.
  - Can call the `migrateLandsToTunnel`, and `migrateQuadsToTunnel` for transferring LAND tokens and quads from the old tunnel contract to the new tunnel contract.

- Can change the admin of the contract with the `changeAdmin` function.
- Can call the `migrateLandsToTunnel`, `migrateToTunnelWithWithdraw`, and `migrateQuadsToTunnel` for transferring LAND tokens and quads from the old tunnel contract to the new tunnel contract.

## Trust Assumptions

The LAND contracts as well as the associated tunnel contracts depend on external contracts to properly function. These external contracts are assumed to be properly implemented and free from bugs. The contracts have the following privileges:

The `LandV3` contract depends on the following external contracts:

- The `operatorFilterRegistry` contract controls what contracts may be approved to transfer tokens as well as which contracts can initiate transfers. This contract can be set to the deployed contract specified in the operator filterer registry. If this contract rejects transfers from contracts that should be able to transfer tokens, specifically if the `_metaTransactionContracts` or LAND tunnel contract are blocklisted, the functionality of the token will be affected.
- The `_metaTransactionContracts` contracts are able to initiate meta-transactions on behalf of accounts with the token contract. It is assumed that these contracts will act faithfully as they are able to execute many sensitive functions including changing the `admin` role, setting `_superOperators`, as well as all transfer and approval-related functions.

The `PolygonLandV2` contract depends on the following external contracts:

- The `operatorFilterRegistry` contract controls what contracts may be approved to transfer tokens as well as which contracts can initiate transfers. This contract can be set to the deployed contract specified in the operator filterer registry. If this contract rejects transfers from contracts that should be able to transfer tokens, specifically if the `_metaTransactionContracts` or LAND tunnel contract are blocklisted, the functionality of the token will be affected.
- The `_trustedForwarder` contract is able to initiate meta-transactions on behalf of accounts with the token contract. It is assumed that this contract will act faithfully as it is able

The `LandTunnelV2` contract depends on the following external contracts:

- The `_trustedForwarder` contract is able to initiate meta-transactions on behalf of accounts. It is assumed that this contract will act faithfully as it is able to execute many sensitive functions including changing the `owner` role, as well as initiating transfers to Polygon.
- The inherited `FxBaseRootTunnel` contract for the fx-portal integration contains external contracts that manage the bridging of tokens between Ethereum and Polygon. It is assumed that the `fxRoot`, `checkpointManager`, and `fxChildTunnel` contracts will be set to the appropriate contracts deployed by Polygon. These contracts must function as intended for tokens to be successfully and properly bridged between Ethereum and Polygon.

The `PolygonLandTunnelV2` contract depends on the following external contracts:

- The `_trustedForwarder` contract is able to initiate meta-transactions on behalf of accounts. It is assumed that this contract will act faithfully as it is able to execute many sensitive functions including changing the `owner` role, as well as initiating transfers to Ethereum.
- The inherited `FxBaseChildTunnel` contract for the fx-portal integration contains external contracts that manage the bridging of tokens between Polygon and Ethereum. It is assumed that the `fxRoot`, and `fxChildTunnel` contracts will be set to the appropriate contracts deployed by Polygon. These contracts must function as intended for tokens to be successfully and properly bridged between Polygon and Ethereum.

The `LandTunnelMigration` and `PolygonLandTunnelMigration` contracts have no dependency on external contracts.

# High Severity

## Potentially unsafe usage of unchecked math

The following instances of unchecked math were identified:

- The operation on line 44 of `LandBaseTokenV3.sol`

- The operation on line 85 of `LandBaseTokenV3.sol`
- The operation on line 228 of `LandBaseTokenV3.sol`
- The operation on line 258 of `LandBaseTokenV3.sol`
- The operation on line 279 of `LandBaseTokenV3.sol`
- The operation on line 293 of `LandBaseTokenV3.sol`
- The operation on line 294 of `LandBaseTokenV3.sol`
- The operation on line 314 of `LandBaseTokenV3.sol`
- The operation on line 317 of `LandBaseTokenV3.sol`
- The operation on line 341 of `LandBaseTokenV3.sol`
- The operation on line 383 of `LandBaseTokenV3.sol`
- The operation on line 390 of `LandBaseTokenV3.sol`
- The operation on line 391 of `LandBaseTokenV3.sol`
- The operation on line 421 of `LandBaseTokenV3.sol`
- The operation on line 486 of `LandBaseTokenV3.sol`
- The operation on line 487 of `LandBaseTokenV3.sol`
- The operation on line 541 of `LandBaseTokenV3.sol`
- The operation on line 543 of `LandBaseTokenV3.sol`
- The operation on line 611 of `LandBaseTokenV3.sol`
- The operation on line 620 of `LandBaseTokenV3.sol`
- The operation on line 40 of `LandV3.sol`

Consider using the latest Solidity version which has built-in math checks against overflows and underflows. Alternatively, consider using the OpenZeppelin `SafeMath` library to protect math operations against overflows and underflows for Solidity versions `>=0.6.0` and `<0.8.0`.

***Update:*** *Resolved in pull request #909 at commit 37b3dfb. The `exists` (`_isValidQuad`) function in the `LandBaseTokenV3` contract has been updated to ensure the input quad size is one of the valid sizes. In addition, upon further review from the OpenZeppelin team, it was noted that unchecked math for both the `LandBaseTokenV3.sol` and `LandV3.sol` does not introduce potential for overflows or underflows. Also, the Sandbox team stated:*

> *We checked for possible cases of `unsafeMath` but feel that we have good validation for the values of parameters, which will never cause a `safeMath` error. We have found one case*

## Medium Severity

### Lack of documentation for complex functionality

The codebase contains several internal functions that perform complex computations but lack sufficient documentation. Further, some state variables have complex implicit assumptions about the values they store that are not documented. This lack of documentation can hinder the maintainability of these functions and variables, making it more challenging for auditors to thoughtfully understand their implications. In particular:

- The `_mintAndTransferQuad`, `_checkAndClearOwner`, `_checkAndClear`, and `_checkBatchReceiverAcceptQuadAndClearOwner` functions in the `PolygonLandBaseTokenV2` and `LandBaseTokenV3` contracts.
- The `_owners` state variable defined in the `ERC721BaseTokenV2` contract in both the Ethereum implementation and Polygon implementation, which is used throughout the `LandBaseTokenV3` and `PolygonLandBaseTokenV2` contracts. This variable uses a complex storage pattern to record ownership of individual LAND tokens as well as "quads". Quads are indexed using a bitmask and the code implicitly assumes that ownership of individual LAND tokens takes precedence over quads. Further, the stored addresses include indicator bits above the 160th bit to mark tokens as burned or have an operator enabled.

To ensure ease of maintainability, consider thoroughly documenting these functions and variables, including both function-level documentation as well as in-line documentation where appropriate.

***Update:*** *Resolved in pull request #916 at commit 3d5acc1.*

### Lack of input id validation in the `_ownerOf` function

The `_ownerOf` function in both Ethereum and Polygon implementations does not validate that the input `id` corresponds to a quad of size 1x1. This allows token ids that correspond to quads of size greater than 1 to be passed. External functions such as `burn`, `approve` and `approveFor` do not expect IDs for quads of such size, leading to unexpected consequences. For example, if a quad with a size greater than 1 were to be burnt, it would lead to the incorrect amount being decremented from `_numNFTPerAddress`.

***Update:*** *Resolved in pull request #921 at commit 97da7fb.*

# Low Severity

## Missing docstrings

The following instances of missing docstrings were identified:

- Lines 11 and 32 in `ERC721BaseTokenV2.sol`
- Line 6 in `LandBaseTokenV3.sol`
- Line 9 in `LandV3.sol`
- Lines 4, 5, 7, 9, 11, 13, 19, 25, 31, 37, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, and 65 in `IOperatorFilterRegistry.sol`
- Line 6 in `MetaTransactionReceiverV2.sol`
- Line 5 in `SuperOperatorsV2`.sol
- Lines 12 and 19 in `ERC721MandatoryTokenReceiver.sol`
- Lines 11 and 17 in `ERC721TokenReceiver.sol`
- Line 3 in `AddressUtils.sol`
- Lines 4, 5, 7, 9, 11, 13, 19, 25, 31, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, and 65 in `IOperatorFilterRegistry.sol`
- Lines 16 and 20 in `ERC2771Handler.sol`
- Line 13 in `ERC721BaseTokenV2.sol`
- Line 7 in `WithAdminV2.sol`
- Line 8 in `WithSuperOperatorsV2.sol`
- Line 6 in `FxBaseChildTunnelUpgradeable.sol`
- Line 10 in `FxBaseRootTunnelUpgradeable.sol`
- Lines 12 and 19 in `IERC721MandatoryTokenReceiver.sol`
- Lines 4, 5, 14, and 23 in `ILandToken.sol`
- Lines 6, 7, 13, 21, and 29 in `ILandTokenV2.sol`
- Lines 6, 13, and 19 in `IPolygonLand.sol`
- Lines 4 and 11 in `IPolygonLandTunnel.sol`
- Lines 6 and 7 in `IPolygonLandWithSetApproval.sol`
- Lines 10, and 195 in `PolygonLandBaseTokenV2.sol`

- Lines 6, 7, and 15 of `IPolygonLandV2.sol`
- Line 38 of `LandTunnelMigration.sol`

Additionally, there are cases that require further completion:

- The return from the `supportsInterface` function in the `LandTunnelV2` contract is undocumented.
- The return from the `supportsInterface` function in the `PolygonLandTunnelV2` contract is undocumented.
- The `_childToken` argument to the `initialize` function in the `PolygonLandTunnelV2` contract is undocumented.

Consider thoroughly documenting all functions (and their parameters) that are part of any contract's public API. Functions implementing sensitive functionality, even if not public, should be clearly documented as well. When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

***Update:*** *Resolved in pull request #907 at commit 9ddf8c9.*

## `require` statements with multiple conditions

The following instances of `require` statements with multiple conditions were identified:

- The `require` statement on line 43 of `LandBaseTokenV3.sol`
- The `require` statement on line 44 of `LandBaseTokenV3.sol`
- The `require` statement on line 140 of `LandBaseTokenV3.sol`
- The `require` statement on line 227 of `LandBaseTokenV3.sol`
- The `require` statement on line 228 of `LandBaseTokenV3.sol`
- The `require` statement on line 420 of `LandBaseTokenV3.sol`
- The `require` statement on line 421 of `LandBaseTokenV3.sol`
- The `require` statement on line 658 of `LandBaseTokenV3.sol`
- The `require` statement on line 38 of `PolygonLandBaseTokenV2.sol`
- The `require` statement on line 39 of `PolygonLandBaseTokenV2.sol`
- The `require` statement on line 61 of `PolygonLandBaseTokenV2.sol`

- The `require` statement on line 103 of `PolygonLandTunnelV2.sol`
- The `require` statement on line 79 of `LandTunnelV2.sol`

To simplify the codebase and raise the most helpful error messages for failing `require` statements, consider having a single `require` statement per condition.

***Update:*** *Resolved in pull request #909 at commit 37b3dfb.*

## Lack of event emission after sensitive actions

The following functions do not emit relevant events after executing sensitive actions.

Ethereum LAND:

- When the `_admin` variable is set in the `initialize` function of the `ERC721BaseTokenV2` contract
- When the filter registry is set in the `LandV3` contract
- When the LAND contract is registered on the Operator Filterer Registry in the `LandV3` contract

Polygon LAND:

- When the `_admin` variable is set in the `initialize` function of the `PolygonLandV2` contract
- When the `_trustedForwarder` is set in the initializer of the `ERC2771Handler` contract
- When the `_trustedForwarder` is set using the `setTrustedForwarder` method in the `PolygonLandV2` contract
- When the filter registry is set in the `PolygonLandV2` contract
- When the LAND contract is registered on the Operator Filterer Registry in the `PolygonLandV2` contract
- When the `_trustedForwarder` is set using the `setTrustedForwarder` method in the `LandTunnelV2` contract

`initialize` function in the `PolygonLandTunnelV2` contract

- When the `admin` is set in the `constructor` of the `PolygonLandTunnelMigration` contract
- When the `admin` is set in the `constructor` of the `LandTunnelMigration` contract

Consider emitting events after sensitive changes occur to facilitate tracking and notify any off-chain clients who may be following the contracts' activity.

***Update:*** *Resolved in pull request #919 at commit c5bb59b.*

## Lack of input validation

Throughout the codebase, there are several functions that lack input validation when changing privileged roles, allowing these roles to be set to the zero address. In particular:

- The `newAdmin` argument of the `changeAdmin` function in the `AdminV2` contract for both implementations
- The `batchTransferQuadToL2` function in the `LandTunnelV2` contract can be called with zero-length arrays for the `sizes`, `xs`, and `ys` arguments. Consider validating that the input arrays have a length of at least 1.
- The `batchTransferQuadToL1` function in the `PolygonLandTunnelV2` contract can be called with zero-length arrays for the `sizes`, `xs`, and `ys` arguments. Consider validating that the input arrays have a length of at least 1.
- The `setLimit` function in the `PolygonLandTunnelV2` contract accepts a quad `size` and the gas limit for that size. The `size` argument is never validated to ensure it corresponds to a valid quad size of 1, 3, 6, 12, or 24.
- The `setMaxLimitOnL1` function in the `PolygonLandTunnelV2` contract does not validate that the input gas limit is non-zero. If the `maxGasOnLimitOnL1` variable is set to zero it would prevent tokens from being transferred to L1 as the gas check within the `batchTransferQuadToL1` function will always fail.

Consider adding a check that prevents setting these roles to the zero address and including a separate function for revoking role rights.

## Operator approval can cause unexpected behavior

When an operator approval is set, the `_owners` data stores a flag at the 256th bit (this is done in both implementations, Ethereum and Polygon).

The `mintAndTransferQuad` function is used by `minters` to mint new quads or to transfer existing quads to a new address. When minting and transferring, the quad may be partially owned by the minter (can own some child quads) with the remainder of the quad being unowned (owned by the zero address). If the minter sets operator approval on any of the LAND tokens within the quad to be minted and transferred, the `mintAndTransferQuad` function will fail. Although the minter owns the token, the check performed on line 357 in the Polygon implementation and on line 264 in the Ethereum implementation will fail since the 256th bit will never match `uint256(uint160(msg.sender))`. As a result, the function call will fail.

Function `_checkBatchReceiverAcceptQuadAndClearOwner` will fail to clear the owner of a quad with an operator approval set, as it also performs ownership checks following the pattern explained above, specifically, in lines 323 and 343 in the Ethereum implementation and in lines 463 and 483 in the Polygon implementation.

Consider disregarding the operator flag when checking ownership within the `_mintAndTransferQuad` and `_checkBatchReceiverAcceptQuadAndClearOwner` functions. Further, if an operator is set for any of the transferred tokens, the operator should be cleared from the operator mapping.

***Update:*** *Resolved in pull request #917 at commit 02f7bc8.*

## Lack of storage gap in upgradeable contracts

Throughout the codebase, there are several contracts that are inherited by upgradeable contracts that do not include a storage gap. In particular:

- The `ERC2771Handler` contract
- The `ERC721BaseTokenV2` contract
- The `WithSuperOperatorsV2` contract

and lead to an incompatible storage layout.

To allow additions of new state variables without compromising the storage compatibility with existing deployments, consider leaving a storage gap at the end of each contract.

***Update:*** *Acknowledged, not resolved. The Sandbox team stated:*

> *Those contracts are already deployed on mainnet, therefore we cannot add gaps without breaking the storage.*

## Setting the root and child tunnel addresses can be front-run

The external `setFxRootTunnel` and `setFxChildTunnel` functions defined in the `FxBaseChildTunnel` and `FxBaseRootTunnel` contracts set the `fxRootTunnel` and `fxChildTunnel` state variables. These contracts are inherited by the `FxBaseChildTunnelUpgradeable` and `FxBaseRootTunnelUpgradeable` contracts and the functions are never overridden. Since these functions do not contain any access control, anyone can call them and set the aforementioned addresses to any address except the zero address. If these functions were front-run, the implementation contracts would need to be redeployed as it will not be possible to change these addresses after they have been set.

Consider either including access control on these functions, calling them within the `initialize` functions, or ensuring that they are called within the same transaction as when the implementation contracts are initialized.

***Update:*** *Resolved in pull request #945 at commit b74749c.*

## Unused variable

The `PolygonLandTunnelMigration` and `LandTunnelMigration` contracts both define a constant `GRID_SIZE` that is not used within these contracts.

Consider removing this variable.

***Update:*** *Resolved in pull request #942 at commit 8a16ac2.*

sets approval for the new land tunnel contract to transfer LAND tokens on behalf of the contract. This function lacks any access control and can be called by anyone.

While this is not directly a security risk as only super-operators are approved to transfer LAND tokens to the contract within the `migrateToTunnelWithWithdraw` function, consider restricting this function such that only the `admin` can call it to further secure the contract.

***Update:*** *Resolved in pull request #946 at commit 90a111e.*

## Notes & Additional Information

### Files specifying outdated Solidity versions

The following instances of files specifying outdated Solidity versions were identified:

- The `pragma` statement on line 2 of `ERC721BaseTokenV2.sol`
- The `pragma` statement on line 2 of `LandBaseTokenV3.sol`
- The `pragma` statement on line 3 of `LandV3.sol`
- The `pragma` statement on line 2 of `OperatorFiltererUpgradeable.sol`
- The `pragma` statement on line 2 of `IOperatorFilterRegistry.sol`
- The `pragma` statement on line 1 of `MetaTransactionReceiverV2.sol`
- The `pragma` statement on line 1 of `SuperOperatorsV2.sol`
- The `pragma` statement on line 1 of `ERC721Events.sol`
- The `pragma` statement on line 1 of `ERC721MandatoryTokenReceiver.sol`
- The `pragma` statement on line 9 of `ERC721TokenReceiver.sol`
- The `pragma` statement on line 1 of `AddressUtils.sol`

Consider taking advantage of the latest Solidity version to improve the overall readability and security of the codebase. Regardless of which version of Solidity is used, consider pinning the version consistently throughout the codebase to prevent bugs due to incompatible future releases.

***Update:*** *Acknowledged, not resolved. The Sandbox team stated:*

> *We decided not to change the solidity version as it causes too many changes.*

- _idInPath of `LandBaseTokenV3.sol`
- _isQuadMinted of `LandBaseTokenV3.sol`
- _getQuadLayer of `LandBaseTokenV3.sol`
- _getQuadById of `LandBaseTokenV3.sol`
- uint2str of `LandV3.sol`
- _isQuadMinted of `PolygonLandBaseTokenV2.sol`
- _getQuadLayer of `PolygonLandBaseTokenV2.sol`
- _idInPath of `PolygonLandBaseTokenV2.sol`
- _getQuadById of `PolygonLandBaseTokenV2.sol`

Consider performing additional testing for the functions above.

***Update:*** *Acknowledged, not resolved. The Sandbox team stated:*

> *Fuzzing testing is not easily possible on the current repository but we're planning to migrate the repository and enable fuzz testing later this year.*

## Lack of indexed event parameters

The following instances could benefit from indexing event parameters:

- line 18 of `LandBaseTokenV3.sol`
- line 10 of `MetaTransactionReceiverV2.sol`
- line 9 of `SuperOperatorsV2.sol`
- line 24 of `PolygonLandBaseTokenV2.sol`
- line 25 of `PolygonLandTunnelMigration.sol`
- line 33 of `PolygonLandTunnelMigration.sol`
- line 27 of `PolygonLandTunnelV2.sol`
- line 23 of `LandTunnelMigration.sol`

Consider indexing event parameters to improve the ability of off-chain services to search and filter for specific events.

***Update:*** *Resolved in pull request #902 at commit 83835cd.*

- `ERC721BaseTokenV2.sol`
- `LandBaseTokenV3.sol`
- `LandV3.sol`
- `MetaTransactionReceiverV2.sol`
- `SuperOperatorsV2.sol`
- `ERC721Events.sol`
- `ERC721MandatoryTokenReceiver.sol`
- `AddressUtils.sol`
- `ERC721TokenReceiver.sol`

To avoid legal issues regarding copyright and follow best practices, consider adding SPDX license identifiers to files as suggested by the Solidity documentation.

***Update:*** *Resolved in pull request #903 at commit dc4bade.*

## Non-explicit imports are used

Non-explicit imports are used throughout the codebase, which reduces code readability and could lead to conflicts between the names defined locally and the ones imported. This is especially important if many contracts are defined within the same Solidity files or if inheritance chains are long.

For instance:

**LAND**

- lines 4-9 of `ERC721BaseTokenV2.sol`
- line 4 of `LandBaseTokenV3.sol`
- lines 5-7 of `LandV3.sol`
- lines 3 and 4 of `MetaTransactionReceiverV2.sol`
- line 3 of `SuperOperatorsV2.sol`
- lines 6-11 of `ERC721BaseTokenV2.sol`
- line 5 of `WithAdminV2.sol`
- lines 5 and 6 of `WithSuperOperatorsV2.sol`

- line 4 of `IPolygonLandV2.sol`

**TUNNEL**

- line 4 of `ILandTokenV2.sol`
- line 4 of `IPolygonLandWithSetApproval.sol`
- lines 4-6 of `PolygonLandTunnelMigration.sol`
- lines 4-11 of `PolygonLandTunnelV2.sol`
- line 4 of `LandTunnelMigration.sol`
- lines 4-9 of `LandTunnelV2.sol`

Following the principle that clearer code is better code, consider using named import syntax (`import {A, B, C} from "X"`) to explicitly declare which contracts are being imported.

***Update:*** *Resolved in pull request #904 at commit 94697a5.*

## Unused imports

The following instances of unused imports were identified:

- Import `ERC721BaseTokenV2` of `LandV3.sol`
- Import `PolygonLandBaseToken` of `PolygonLandTunnelV2.sol`

Consider removing unused imports to improve the overall clarity and readability of the codebase.

***Update:*** *Resolved in pull request #905 at commit 4736d01.*

## Unused named return variables

Throughout the codebase there are multiple instances of unused named return variables:

- The `isOperator` return variable in the `isApprovedForAll` function in `ERC721BaseTokenV2.sol`.
- The `trustedForwarder` return variable in the `getTrustedForwarder` function in `ERC2771Handler.sol`.

- The `sender` return variable in the `_msgSender` function in `ERC721BaseTokenV2.sol`.
- The `sender` return variable in the `_msgSender` function in `PolygonLandTunnelV2.sol`.
- The `sender` return variable in the `_msgSender` function in `PolygonLandV2.sol`.
- The `sender` return variable in the `_msgSender` function in `LandTunnelV2.sol`.

Consider either using or removing any unused named return variables.

***Update:*** *Resolved in pull request #906 at commit 697bd41.*

## Lack of EIP-173 support for operator filter registry

The OpenSea operator filter registry lets a smart contract manage the operators allowed to transfer tokens on behalf of users. If the contract implementing the registry follows EIP-173, the `owner` is able to manage the registry on behalf of the contract. The `PolygonLandV2` contract does not follow EIP-173 but instead has an equivalent `admin` role.

To enable easier control over the registry and provide access to more functionality without requiring an upgrade to the token contract, consider implementing EIP-173.

***Update:*** *Acknowledged, not resolved. The Sandbox team stated:*

> *We decided not to implement the EIP-173 as the `OperatorFilterSubscription` contract will handle the administration.*

## Redundant code

Consider making the following changes to eliminate redundant code:

Ethereum LAND:

- The `_register` function of `OperatorFiltererUpgradeable` contract in line 24 handles the case where `subscriptionOrRegistrantToCopy` is the zero address, but this path will never be executed as there is a `require` statement validating that

function in the `LandBaseTokenV3` contract is unnecessary.

- The `_checkTransfer` function in the `ERC721BaseTokenV2` contract will either revert or return `true`. Returning `true` from this function is unnecessary.

Polygon LAND:

- The `_register` function of the `OperatorFiltererUpgradeable` contract in line 27 handles the case where `subscriptionOrRegistrantToCopy` is the zero address, but this path will never be executed as there is a `require` statement validating that `subscriptionOrRegistrantToCopy != address(0)` in the caller function. Consider removing this code path.
- Checking if the output of the `exists` function is `true` in the `mintAndTransferQuad` function in the `PolygonLandBaseTokenV2` contract is unnecessary.
- The `_checkTransfer` function in the `ERC721BaseTokenV2` contract will either revert or return `true`. Returning `true` from this function is unnecessary.
- When calling the `safeTransferFrom` function in the `PolygonLandV2` contract without the `data` argument, the `onlyAllowedOperator` modifier is called twice: once by the `safeTransferFrom` function without the `data` argument, and again by the `safeTransferFrom` function with the `data` argument.

***Update:*** *Partially resolved in pull request #910 at commit 28607d2. The following instances remain unresolved:*

- Ethereum LAND: The `_register` function of `OperatorFiltererUpgradeable` contract in line 24 handles the case where `subscriptionOrRegistrantToCopy` is the zero address, but this path will never be executed as there is a `require` statement validating that `subscriptionOrRegistrantToCopy != address(0)` in the caller function. Consider removing this code path.

- Polygon LAND: The `_register` function of the `OperatorFiltererUpgradeable` contract in line 27 handles the case where `subscriptionOrRegistrantToCopy` is the zero address, but this path will never be executed as there is a `require` statement

In addition, The Sandbox team stated:

> *We decided not to resolve this issue to keep the code as close as the original one from OpenSea.*

### Reuse `onlyAdmin` modifier

The `onlyAdmin` modifier can be used instead of the `require` checks:

- In the `setMinter` function in the `PolygonLandBaseTokenV2` contract
- In the `setSuperOperator` function in the `WithSuperOperatorsV2` contract

To improve the readability of the codebase, consider using the `onlyAdmin` modifier instead of the `require` checks.

***Update:*** *Resolved in pull request #911 at commit 6fe419b.*

### Trusted forwarder validated against operator filter registry

The trusted forwarder is used throughout the contract to enable meta-transactions. With the addition of the operator filter registry, any transaction initiated by the trusted forwarder will validate the trusted forwarder against the registry as the `onlyAllowedOperator` modifier checks `msg.sender` rather than `_msgSender`.

Consider using `_msgSender` within the `onlyAllowedOperator` modifier to reduce gas consumption when performing meta-transactions.

***Update:*** *Resolved in pull request #924 at commit 967f133.*

### Typographical errors

Consider addressing the following typographical errors.

Ethereum LAND:

- On line 8 of `OperatorFiltererUpgradeable.sol`, "*subscibe*" should be "*subscribe*", and "or copy *or just to* the subscription provided" should be "or copy the subscription

- On line 470 of `LandBaseTokenV3.sol`, "*transfered*" should be "*transferred*".
- On line 485 of `LandBaseTokenV3.sol`, "*itereates*" should be "*iterates*".
- On line 513 of `LandBaseTokenV3.sol`, "*ittereated*" should be *iterated*.
- On line 81 of `ERC721BaseTokenV2.sol`, "*resset*" should be "*reset*".
- On line 81 of `ERC721BaseTokenV2.sol`, "*overriden*" should be "*overridden*".

Polygon LAND:

- On line 8 of `OperatorFiltererUpgradeable.sol`, "*subscibe*" should be "*subscribe*".
- On line 8 of `OperatorFiltererUpgradeable.sol`, "or copy *or just to* the subscription provided" should be "or copy the subscription provided".
- On line 38 of `LandTunnelV2` contract, "*trasnfer*" should be "*transfer*".
- On line 51 of `LandTunnelV2` contract, "*trasnfer*" should be "*transfer*".
- On lines 54-57 of the `PolygonLandTunnelMigration` contract, "cant" should be "can't".
- On line 142 of `PolygonLandTunnelV2` contract, "*trasnfer*" should be "*transfer*".
- On line 155 of `PolygonLandTunnelV2` contract, "*trasnfer*" should be "*transfer*".
- On line 86 of `ERC721BaseTokenV2.sol`, "*send*" should be "*sender*".
- On line 155 of `ERC721BaseTokenV2.sol`, there should be a space after "*token*".
- On line 355 of `ERC721BaseTokenV2.sol`, "*adddress*" should be "*address*".
- On line 373 of `ERC721BaseTokenV2.sol`, "*adddress*" should be "*address*".
- On line 418 of `ERC721BaseTokenV2.sol`, "*recieving*" should be "*receiving*".
- On line 625 of `PolygonLandBaseTokenV2.sol`, "*transfered*" should be "*transferred*".
- On line 640 of `PolygonLandBaseTokenV2.sol`, "*itereates*" should be "*iterates*".
- On line 664 of `PolygonLandBaseTokenV2.sol`, "*qua*" should be "*quad*".
- On line 668 of `PolygonLandBaseTokenV2.sol`, "*ittereated*" should be "*iterated*".
- On line 163 of `PolygonLandV2.sol`, the double quote `""` after "true" should be a single quote `'`.

***Update:*** *Resolved in pull request #912 at commit e11a210.*

## Unclear event names

Ethereum LAND:

- The `MetaTransactionProcessor` event defined in the `MetaTransactionReceiverV2` contract
- The `Minter` event defined in the `LandBaseTokenV3` contract
- The `SuperOperator` event defined in the `SuperOperatorsV2` contract

Polygon LAND:

- The `Minter` event defined in the `PolygonLandBaseTokenV2` contract
- The `SuperOperator` event defined in the `WithSuperOperatorsV2` contract

Consider renaming these events using descriptive names that provide a clear context of their intended purpose.

***Update:*** *Acknowledged, not resolved. The Sandbox team stated:*

> *We decided not to change the event names because those events are already consumed.*

## Lack of attribution

Throughout the codebase, there are files that have been copied and modified from the OpenSea operator filter registry codebase. The original contracts use the MIT license which requires attribution. In particular:

Ethereum LAND:

- OperatorFiltererUpgradeable.sol
- IOperatorFilterRegistry.sol

Polygon LAND:

- OperatorFiltererUpgradeable.sol
- IOperatorFilterRegistry.sol

## Gas optimizations

The following opportunities for gas optimization were identified:

Ethereum LAND:

- In the `_checkTransfer` function of the `ERC721BaseTokenV2` contract, lines 150 and 151 perform the same check. Consider refactoring the function to avoid duplicated checks.
- Using a bitmask to clear the highest 8 bits in the `_getX` and `_getY` functions in the `LandBaseTokenV3` contract would be more gas efficient.

Polygon LAND:

- Using a bitmask to clear the highest 8 bits in the `_getX` and `_getY` functions in the `PolygonLandBaseTokenV2` contract would be more gas efficient.

Consider optimizing gas consumption by adjusting the aforementioned issues.

***Update:*** *Resolved in pull request #915 at commit 2c0b610.*

## Inconsistent use of named return values

The following contracts contain some functions with named return values, and some without.

In Ethereum LAND:

- The `LandBaseTokenV3` contract
- The `AddressUtils` library
- The `ERC721BaseTokenV2` contract

In Polygon LAND:

- The `PolygonLandV2` contract
- The `PolygonLandBaseTokenV2` contract
- The `ERC721BaseTokenV2` contract
- The `ERC2771Handler` contract

Consider using named return values consistently throughout every contract and library to provide a clear understanding of the code's behavior.

***Update:*** *Resolved in pull request #918 at commit ef05b41.*

## Naming issues hinder code understanding and readability

Throughout the codebase, there are several functions and variables that could be renamed to better reflect their purpose, in particular:

Ethereum LAND:

- `_checkAndClear` function of `LandBaseTokenV3` contract.
- The `landMinted` variable in the `_mintAndTransferQuad` function in the `LandBaseTokenV3` contract corresponds to the number of LAND tokens transferred, not minted.

Polygon LAND:

- The `_checkAndClear` function in the `PolygonLandBaseTokenV2` contract.
- The `landMinted` variable in the `_mintAndTransferQuad` function in the `PolygonLandBaseTokenV2` contract corresponds to the number of LAND tokens transferred, not minted.
- The `maxAllowedQuads` state variable in the `PolygonLandTunnelV2` contract corresponds to the maximum amount of LAND tokens, not quads.
- The `user` address parameter in the `mintQuad` function of the `IPolygonLand` interface should be `to` address
- In `PolygonLandTunnelV2` contract, `gasLimit` should be `totalGasLimit`
- The naming of `setLimit` and `setupLimits` functions in the PolygonLandTunnelV2 contract is unclear and confusing.

Consider renaming these functions and variables to improve the codebase's readability.

***Update:*** *Partially resolved in pull request #922 at commit 0de8738. The Sandbox team stated:*

> __checkAndClearOwner called during the mintandtransfer to check if the sub quads are owned by the msg.sender and clears it_
> __checkBatchReceiverAcceptQuadAndClearOwner checks if to in the mintandtransfer if is a contract and can handle ERC721(onERC721 receive functions) and clears the owner of 1x1 land for every 1x1 land in the Quad to be mint and transfer_
> __numLandMinted is the cumulative value of number of land tokens that are found to be already minted. numLandMinted variable correspond to number of Land already minted_

### `public` function that should have `external` visibility

The following `public` function should be `external`:

- The `batchTransferQuadToL2` function in the `LandTunnelV2` contract.

Consider changing the visibility of this function to `external` in order to clarify that this function will only be called by external contracts.

***Update:*** *Resolved in pull request #947 at commit 0d38cb3.*

### Inconsistent ordering of functions

The codebase generally follows the recommended order in the Solidity Style Guide, however there are some instances where contracts deviate from the style guide. In particular:

- In the `LandTunnelMigration` contract there are functions defined before the `constructor`.
- In the `PolygonLandTunnelMigration` contract there are functions defined before the `constructor`.
- In the `PolygonLandTunnelV2` contract there are functions defined before the `initialize` function. While not directly part of the Solidity style guide, the `initialize` function should be considered comparable to a `constructor`.

To improve the project's overall legibility, consider standardizing ordering throughout the codebase, as recommended by the Solidity Style Guide.

Throughout the codebase, there are several variables that could be `immutable`. For instance:

- The `polygonLand` variable in the `PolygonLandTunnelMigration` contract.
- The `newLandTunnel` variable in the `PolygonLandTunnelMigration` contract.
- The `oldLandTunnel` variable in the `PolygonLandTunnelMigration` contract.
- The `landToken` variable in the `LandTunnelMigration` contract.
- The `newLandTunnel` variable in the `LandTunnelMigration` contract.
- The `oldLandTunnel` variable in the `LandTunnelMigration` contract.

To better convey the intended use of variables and to potentially save gas, consider adding the `immutable` keyword to variables that are only set in the constructor.

***Update:*** *Resolved in pull request #953 at commit 6b5b33f.*

## Unused bytes `data` value

Throughout the codebase, there are calls for transferring LAND tokens that include the `data` argument. These calls pass a value of `"0x"` for the `data` argument which is never used within the calls. In particular:

- Within the `migrateLandsToTunnel` function in the `LandTunnelMigration` contract.
- Within the `migrateQuadsToTunnel` function in the `LandTunnelMigration` contract.
- Within the `migrateLandsToTunnel` function in the `PolygonLandTunnelMigration` contract.
- Within the `migrateToTunnelWithWithdraw` function in the `PolygonLandTunnelMigration` contract.
- Within the `migrateQuadsToTunnel` function in the `PolygonLandTunnelMigration` contract.

Consider passing empty bytes data ( `""` ) to these functions as the input value is never used and consumes unnecessary gas.

The codebase contains several contracts that are intended to be deployed as the implementations for a proxy pattern. These implementation contracts contain `initialize` functions that replace a `constructor` for initializing the proxy contract state. To ensure the `initialize` function is only executed once, the `initializer` modifier from the inherited OpenZeppelin `Initializable` contract is added. Since there is no access control on the `initialize` function, anyone is able to call this function on the implementation contract with arbitrary input arguments. Consider calling the `_disableInitializers` function, introduced in version 4.6.0 of the OpenZeppelin contracts library, from the constructors of the implementation contracts to prevent them from being initialized. In particular, the following implementation contracts should be updated:

- `PolygonLandV2`
- `LandTunnelV2`
- `PolygonLandTunnelV2`

While not a direct security concern, it is a good practice to prevent the implementation contract from being initialized as this could allow an attacker to take over the contract. This would not affect the functionality of the proxy contract as only the storage of the implementation contract would be affected.

***Update:*** *Acknowledged, not resolved. The Sandbox team stated:*

> *We decided not to fix this issue as it would require upgrading the OpenZeppelin contracts to 4.6.0, which is not an option for us with the current implementation.*

No critical severity issues were found. One (1) high, two (2) medium and nine (9) low severity issues and 22 notes were reported mainly addressing improvement opportunities to the overall quality of the codebase.

## Monitoring Recommendations

While audits help in identifying code-level issues in the current implementation and potentially the code deployed in production, The Sandbox team is encouraged to consider incorporating monitoring activities in the production environment. Ongoing monitoring of deployed contracts helps identify potential threats and issues affecting production environments. With the goal of providing a complete security assessment, the monitoring recommendations section raises several actions addressing trust assumptions and out-of-scope components that can benefit from on-chain monitoring.

Some of the provided recommendations are affected by how the Polygon Fx-Tunnel bridge is implemented. Specifically, when the Polygon Bor block producer calls the `processMessageFromRoot` function in the `FxBaseChildTunnel` contract, it is executed as a system call and does not produce a transaction receipt or emit events. This affects the `PolygonLandTunnelV2` contract specifically as the `Deposit` event in the `_syncDeposit` function will not be emitted nor will any `Transfer` events emitted by the `mintAndTransferQuad` call to the `childToken`.

These events are however emitted within special state-sync "blocks" that are not a part of the blockchain and can be retrieved using the `eth_getLogs` endpoint by filtering for a blockhash of the format `keccak256("matic-bor-receipt-" + block number + block hash)`.

Ensure that all monitoring activities that rely on these events monitor the correct blockhashes that include them.

## Token

**Critical:** When LAND tokens or quads of LAND tokens are transferred between accounts, the `balanceOf` for those accounts should be appropriately incremented/decremented. When transferring quads, the amount the balance changes by should be the total number of LAND tokens within the quad. Consider monitoring accounts balances before and after transfers with the `LandV3` and `PolygonLandV2` contracts by watching for the `Transfer(address, address, uint256)` events.

parent quads of the transferred quad should still have no owner or be owned by the new owner of the transferred quad. Consider monitoring for transfers of quads by the `transferQuad`, `batchTransferQuad`, and `mintAndTransferQuad` functions within the `LandV3` and `PolygonLandV2` contract to ensure the correct ownership after a transfer.

**Critical:** There are multiple privileged actions with serious security implications:

- The admin of the `LandV3` and `PolygonLandV2` contracts controls access to actions such as updating the operator filterer registry, adding and removing minters, and setting the meta transaction processor. Changes to the admin roles can be monitored by watching for the `AdminChanged(address, address)` event.
- The super operators in the `LandV3` and `PolygonLandV2` contracts can set arbitrary approvals and make arbitrary transfers. Changes to the super operators can be monitored by watching for the `SuperOperators(address, bool)` event.
- The owner of the `LandTunnelV2` and `PolygonLandTunnelV2` contracts controls setting gas limits, the maximum amount of quads that can be transferred in a single transaction, the address of the trusted forwarder, and can pause and unpause the contract. Changes to the owner roles can be monitored by watching for the `OwnershipTransferred(address, address)` event.
- The admin of the `LandTunnelMigration` and `PolygonLandTunnelMigration` contracts controls transferring LAND tokens between the old tunnel contract and the new tunnel contract. Changes to this role can be monitored by watching for the `AdminChanged(address, address)` event.
- Minters in `LandV2` and `PolygonLandV2` contracts can mint new LAND tokens and quads. Changes to this role can be monitored by watching for the `Minter(address, bool)` events.

**High:** When LAND tokens are burned using the `burn` function or `burnFrom` function in the `LandV3` and `PolygonLandV2` contracts, they should never be recoverable. Consider monitoring for transfers of burned tokens as this could indicate the contracts are not functioning as expected.

**Low:** When quads are transferred in the `LandV3` and `PolygonLandV2` contracts using the `mintQuad`, `mintAndTransferQuad`, `transferQuad`, or `batchTransferQuad`

missing event emission may indicate that a token has not been transferred properly.

Consider monitoring triggers of these administrator functions to ensure all changes are expected. Further, consider monitoring the accounts with these roles to ensure there is no suspicious activity within the accounts that could suggest the accounts/contracts have been compromised.

## Technical

**Critical:** When the LAND tunnel is used to bridge tokens from Ethereum to Polygon, the tunnel contract on Ethereum will hold the bridged LAND tokens while the user who bridged the tokens will have control over the corresponding LAND tokens on Polygon. Similarly, when bridging from Polygon to Ethereum, either the tokens are locked in the Polygon tunnel contract while the owner has full control over them on Ethereum, or they are yet to be minted on Polygon if they have never been bridged to Polygon. Consider monitoring the tunnel contracts to ensure that all tokens not held by the tunnel contracts on Ethereum/Polygon are either held by the tunnel contracts on Polygon/Ethereum, or have not yet been minted. If there token `id` that is not held by the either bridge contract on Ethereum or on Polygon, then the owner(s) of that token on Ethereum and Polygon would be able to freely transfer the same token. This would introduce an issue where the same token is being used on different chains and would prevent the token from being bridged in the future.

**Critical:** When a `Withdraw(address, uint256, uint256, uint256, bytes)` event is emitted by the `PolygonLandTunnelV2` contract on Polygon, there must have been a corresponding `Deposit(address, uint256, uint256, uint256, bytes)` event into the `LandTunnelV2` contract on Ethereum and the token that was withdrawn on Polygon must be held by the tunnel contract on Ethereum. Similarly, when a `Withdraw(address, uint256, uint256, uint256, bytes)` event is emitted by the `LandTunnelV2` contract on Ethereum, there must have been a corresponding `Deposit(address, uint256, uint256, uint256, bytes)` event into the `PolygonLandTunnelV2` contract on Polygon and the token that was withdrawn on Ethereum must be held by the tunnel contract on Polygon. Consider monitoring to ensure these properties hold. When monitoring for the `Withdraw(address, uint256, uint256, uint256, bytes)` event on Polygon, the event will not be emitted as the transaction is run as a system call. Refer to the Polygon documentation for how these events can be retrieved.

are valid. That is, given a quad at coordinates `(x,y)` with size `size`, the following properties must hold:

- `x mod size == 0` and `y mod size == 0`
- `size in {1, 3, 6, 12, 24}`
- `0 <= x <= 408 - size` and `0 <= y <= 408 - size`

**High:** The meta transaction processor role in the `LandV3` contract and the trusted forwarder role in the `PolygonLandV2` contract allow meta-transactions to be performed with these contracts. As contracts with this privilege can perform critical actions on behalf of users, any unexpected changes to these roles could signal an attack attempt. Consider monitoring for the `MetaTransactionProcessor(address, bool)` event in the `LandV3` contract, and for calls to the `setTrustedForwarder` function in the `PolygonLandV2` contract.

**High:** Many functions on the LAND token contracts should only be callable for 1x1 quads. In particular:

- `approveFor`
- `approve`
- `transferFrom`
- `safeTransferFrom`
- `batchTransferFrom`
- `safeBatchTransferFrom`
- `setApprovalForAll`
- `setApprovalForAllFor`
- `burn`
- `burnFrom`

If any of these functions for the `LandV3` and `PolygonLandV2` contracts are successfully executed with token `id` values that represent quads larger than 1x1, this could signal improper operator of the contracts.

re-executed. Consider monitoring pending withdrawals on Polygon to ensure that their execution is successful.

**Medium:** Any changes to the operator filterer configuration for the `LandV3` and `PolygonLandV2` contracts could impact token transfers. Specifically, if the trusted forwarder is blacklisted by the operator filterer, meta transactions with the contract will fail. Additionally, if any exchange that holds a large quantity of LAND tokens is blocklisted, this could impact users expecting to list their LAND tokens on the exchange. Changes to the configuration can be monitored by watching for calls to the `register` and `setOperatorRegistry` functions. Further, changes to the configured `operatorFilterRegistry` can be monitored by watching for the appropriate events on the configured contract.

## Suspicious activity

**Critical:** The `LandTunnelV2` and `PolygonLandTunnelV2` contracts contain functionality to pause the contracts which prevents tokens from being bridged from Ethereum to Polygon or back. Only the owner of these contracts should be able to call the associated functions to pause/unpause. Consider monitoring for unexpected `Paused(address)` and `Unpaused(address)` events to ensure the system is functioning as expected.

**Medium:** Consider monitoring for unexpectedly large LAND transfers via the `Transfer(address, address, uint256)` event in the `LandV3` and `PolygonLandV2` contracts as this could signal a potential attack attempt.


## Related Posts

Beefy

BRUSHFAM
OpenBrush Contracts

Linea

## Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

## OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

## Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits