Quantstamp Security Assessment Certificate

May 19th 2022 — Quantstamp Verified

Sandbox

This audit report was prepared by Quantstamp, the leader in blockchain security.

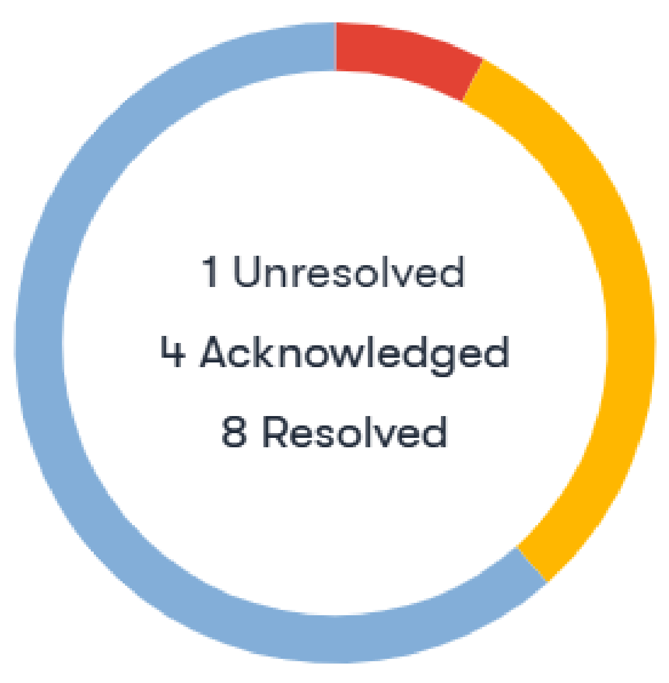
Executive Summary

Type	Decentralized Gaming Platform
Auditors	Souhail Mssassi, Research Engineer David Knott, Senior Research Engineer Philippe Dumonet, Senior Research Engineer
Timeline	2022-03-22 through 2022-05-19
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	[Land Bridge](#) [Sandbox Documentation](#)
Documentation Quality	Medium
Test Quality	High

Source Code

Repository	Commit
sandbox-smart-contracts	8831c95
sandbox-smart-contracts-private	36efd82

Total Issues	**13**	(8 Resolved)
High Risk Issues	**3**	(2 Resolved)
Medium Risk Issues	0	(0 Resolved)
Low Risk Issues	**5**	(2 Resolved)
Informational Risk Issues	**3**	(3 Resolved)
Undetermined Risk Issues	**2**	(1 Resolved)



⌃⌃ High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
⌃ Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
⌄ Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
○ Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
? Undetermined	The impact of the issue is uncertain.

● Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
● Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
● Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
● Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Initial audit:
Through reviewing the code, we found **13 potential issues** of various levels of severity:3 high-severity, 5 low-severity, 3 informational-severity, and 2 undetermined issues. We recommend addressing all the issues before deploying the code in production.

ID	Description	Severity	Status
QSP-1	Impermanent Burns	High	Fixed
QSP-2	Centralization Risk	High	Acknowledged
QSP-3	Land Quads Can Be Double-Spent	High	Fixed
QSP-4	ERC721 Token Contracts Not Able To Accept Batches	Low	Unresolved
QSP-5	Fixed Stored Metadata	Low	Acknowledged
QSP-6	Bridge Contracts Always Accept ERC721 Token	Low	Fixed
QSP-7	ERC2771 Meta Transactions Not Supported For Admin	Low	Acknowledged
QSP-8	Possible Inconsistency In `maxAllowedQuads`	Low	Mitigated
QSP-9	Renounceable Admin/Ownership	Informational	Fixed
QSP-10	Pragma Unlocked	Informational	Fixed
QSP-11	Unused Interface	Informational	Fixed
QSP-12	Incomplete Pauses	? Undetermined	Acknowledged
QSP-13	Burned Quads Cannot Be Minted Again	? Undetermined	Fixed

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

1. Installed the Slither tool: `pip install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Impermanent Burns

Severity: High Risk

Status: Fixed

File(s) affected: `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`, `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol`

Description: `ERC721BaseToken::L337(_burn)` uses the `BURN_FLAG` to flip a bit, denoting whether or not a given 1x1 quad has been burned. The bit flip is impermanent, though, as `PolygonLandBaseToken::L518(_checkAndClear)` overwrites it, allowing quads larger than 1x1 that contain a burnt 1x1 quad to be transferred, unburning any burnt 1x1s.
Similarly, `PolygonLandBaseToken::L555(_ownerAndOperatorEnabledOf)`'s checks whether or not a given 1x1 quad has been burnt, but does not do anything if the check evaluates to `true`. This enables `ERC721BaseToken::L156(burnFrom)`, `ERC721BaseToken::L189(getApproved)`, `ERC721BaseToken::L285(_batchTransferFrom)`, and `ERC721BaseToken::L411(_checkTransfer)` which rely on `_ownerAndOperatorEnabledOf` for ownership validation to operate as if a burnt 1x1 quad has not been burned. In practice, burnt 1x1 quads can be burnt multiple times and unburnt via larger quad transfers.

Recommendation: Modify `_ownerAndOperatorEnabledOf` to return a zero address if an `owner`'s `BURNT_FLAG` bit is set to `true`.

Update: `_ownerAndOperatorEnabledOf` now returns the zero address if an `owner`'s `BURNT_FLAG` bit is set to true. `_checkAndClear` was also modified to check whether a 1x1 quad has been burned prior to setting the 1x1 quad's owner value to zero which previously overwrote the `BURNT_FLAG`.

QSP-2 Centralization Risk

Severity: High Risk

Status: Acknowledged

File(s) affected: `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandV1.sol`, `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`, `src/solc_0.8/polygon/root/land/LandTunnel.sol`

Description: The `ERC721BaseToken` tracks a list of so-called "super operators". These "super operators" can be assigned and removed by the contract admin. The "super operators" have the ability to transfer and approve ERC721 token transfers on behalf of any address at any time. There is no limitation or opt-out to this ability.
The `PolygonLandV1` contract also has an admin address which has the ability to adjust the `polygonLandTunnel` address, which determines which contract address is allowed to mint polygon land tokens. A misuse of this ability may lead to the admin minting unbacked land tokens on the Polygon Land token, polygon.
In the `PolygonLandTunnel` contract, the `Ownable` contract is used together with the `Pausable` contract. The `PolygonLandTunnel` contract allows the `owner` address to "pause" the contract. This pause effectively prevents users from bridging their land tokens back from the Polygon Land token to their original L1 versions. This feature gives the owner address the power to block redemptions of Polygon Land tokens, making them effectively worthless. If misused, this ability could lead to severe loss for Polygon Land token users.
Furthermore, beyond "pausing" the bridge, the `PolygonLandTunnel` owner address can also set an arbitrary ERC2771 forwarder, meaning they could bridge the bridged Polygon Land tokens of any address which has approved the bridge contract. This could be done by setting a custom ERC2771 forwarder contract, which allows the owner to imitate any address or wallet.
The `LandTunnel` contract implements the same privileges as the `PolygonLandTunnel` contract, allowing the owner address (`owner`) to "pause" the contract and update the ERC2771 forwarder.

Recommendation: It is recommended that these abilities either be removed or strongly mediated (e.g., via decentralized governance). If these features are to be kept and operated in a centralized manner, end-user documentation should be added informing users of the associated risks.

Update: The team acknowledged the issue, stating that they "have been aware of it".

QSP-3 Land Quads Can Be Double-Spent

Severity: High Risk

Status: Fixed

File(s) affected: `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandV1.sol`

Description: The `PolygonLandBaseToken` contract implements a system of land quadrants aka "quads" in sizes of 1x1, 3x3, 6x6, 12x12 and 24x24. These quads are meant to be transferable and divisible. An owner of a 12x12 quad for example is meant to be able to transfer any contained 6x6, 3x3 or 1x1.
However, due to a flaw in the current implementation, transferring a sub-quad of size 3x3 or larger does not completely bind the ownership of that quad to the recipient. Besides the new child-quad owner, the previous parent-quad owner still has the ability to transfer the sub-quad.
This issue arises from the logic pattern implemented in the `_regroupNxN` methods. As long as no child-quads have been moved, the `_regroupNxN` methods will always revert to checking for parent-quad ownership if the caller is not the owner of the direct quad.
Note: After additional review, it has been determined that the ability for larger quad owners to transfer child quads they no longer own is also present in the currently deployed mainnet Sandbox LAND contract (implementation) and should be addressed prior to the publishing of this report.

Exploit Scenario:

1. The owner of a 6x6 quad sells a 3x3 quad.
2. The 6x6 owner calls `transferQuad` and `_regroup3x3` sets the 3x3 quad's owner to the buyer's address.
   - The 6x6 owner's quad is not subdivided, so the sold 3x3 quad has two owners.
3. The 6x6 quad call owner calls `transferQuad`, transferring ownership of the 3x3 quad they sold back to themselves, effectively stealing land from the 3x3 buyer.
   - This is possible because having ownership of a super-quad, in this case, passing the 6x6 quad check on `PolygonLandBaseToken::L405(_regroup3x3)` is sufficient to transfer ownership of a sub-quad.

Recommendation: Besides not being fully functional, the minting and transfer of quads is quite inefficient as all sub-quads are always checked. Therefore, a complete overhaul of the quad system is recommended. While treating larger quads differently than small quads may have small advantages, it is also recommended to change this design aspect. Allowing larger quads to be transferred via the ERC721 standard defined transfer methods (`transferFrom(address,address,uint256)`, `safeTransferFrom(address,address,uint256)`, `safeTransferFrom(address,address,uint256,bytes)`) would allow for larger quads to directly be listed and sold on traditional NFT marketplaces such as OpenSea. Regardless of whether the quad system is overhauled or not, it is recommended that more tests be implemented so that such issues may be caught sooner in future iterations.
If the quad logic is to be overhauled, one potential design approach is to never allow a child quad to be directly transferred from an owner who only owns a parent quad. Have separate methods for merging, splitting and transferring quads. This not only simplifies the logic, but makes the implementation less complex. Splitting very large quads into smaller quads and transferring a smaller piece can be achieved in a single transaction by leveraging a `Multicall` library. Furthermore, when minting quads on Polygon Land token checking all sub-quads can be avoided as

mints will only be triggered via the bridge and can therefore simply be trusted, assuming the issue with the mainnet land token contract has been resolved.
If a sanity check to prevent duplicate mints is still desired, this still does not require a full check of all sub-quads. Instead, looking at quads as a tree structure, ensure that minting of sub-quads sets a "reserved" flag in all its parent quads. This way, when new quads are being minted, the contract can simply check if the quad to be minted is already reserved or owned and also check top-down that its parent quads have not been minted yet. While this comes at an increased cost for the first child quad within a parent quad, overall the gas cost will be strongly reduced, especially for larger quads as only its parent quads will have to be checked instead of all children.

Update: The Sandbox team broke out parent quad ownership checks into a seperate function, `_ownerOfQuad` and fixed the broken checks that allow for quads to be double spent.

QSP-4 ERC721 Token Contracts Not Able To Accept Batches

Severity: Low Risk

Status: Unresolved

File(s) affected: `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandV1.sol`

Description: Contracts which comply with the ERC721 standard and implement the `onERC721Received` method may not be able to accept ERC721 tokens sent to them as part of a batch. This is due to the `ERC721BaseToken` contract expecting a custom `onERC721BatchReceived` check. While having a single checking method may allow for validation to be more efficient, requiring it will prevent compliant contracts from accepting tokens sent to it in a batch.

Recommendation: It is recommended that the ERC165 standard be used to check whether receiving contracts support the special batch accepting procedure, and revert to repeated calls to the standard `onERC721Received` method if `onERC721BatchReceived` is not supported. This approach allows specific contracts to directly validate batched transfers while still supporting normal ERC721 compliant receiving contracts.

Update: The Sandbox team states that they "added reverted calls to `onERC721Received` if `onERC721BatchReceived` is not supported." However, no code changes involving how the audited contracts deal with `onERC721BatchReceived` not being supported were found.

QSP-5 Fixed Stored Metadata

Severity: Low Risk

Status: Acknowledged

File(s) affected: `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandV1.sol`

Description: The metadata retrieval method `tokenURI` returns an HTTP-based metadata address where the parent domain is fixed. This can be problematic due to two factors:

- Centrally stored: Metadata stored on a specific domain is subject to being changed by the domain owner and introduces a central failure point.
- Fixed: The fixing of URL structure to a specific parent domain disallows future upgrades. This may be important in the case of, e.g., re-branding efforts or if progressive decentralization is desired.

Recommendation: It is recommended that token asset metadata be changeable to allow for future flexibility. Furthermore, it is also recommended that asset metadata be stored immutably and in a decentralized way whenever possible to remove central points of failure and the risk of metadata loss.

Update: The Sandbox team states that they "have been aware of it. Needs no change."

QSP-6 Bridge Contracts Always Accept ERC721 Token

Severity: Low Risk

Status: Fixed

File(s) affected: `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`, `src/solc_0.8/polygon/root/land/LandTunnel.sol`

Description: The `PolygonLandTunnel` contract implements the `onERC721Received` method as described by the ERC721 standard. While this method is required to accept ERC721 tokens from the Polygon Land token contract, the current implementation also accepts any arbitrary ERC721 token contract, including land tokens transferred to it outside the `batchTransferQuadToL1` bridging method.
Similarly to the `PolygonLandTunnel` contract, the `LandTunnel` contract also has this issue.
While this issue relies on user error, it is best to mitigate such issues when possible because unfortunately user error is quite common. A user mistakenly transferring their NFT to the bridge would result in the token being permanently stuck.

Recommendation: Implement a flag that can be set at the start of the `batchTransferQuadToL{1/2}` method and unset on completion. The state of the flag should then be checked in the `onERC721Received` and `onERC721BatchReceived` methods. This will prevent the contracts from accepting land or other ERC721 NFT transfers outside the `batchTransferQuadToL{1/2}` bridging methods. This ensures that a user cannot accidentally send land tokens to the contract without first approving and triggering a proper bridging action.

Update: The Sandbox team added checks to only accept ERC721 tokens when bridge transferring is occuring.

QSP-7 ERC2771 Meta Transactions Not Supported For Admin

Severity: Low Risk

Status: Acknowledged

File(s) affected: `src/solc_0.8/common/BaseWithStorage/WithSuperOperators.sol`, `src/solc_0.8/common/BaseWithStorage/WithAdmin.sol`

Description: The `WithSuperOperators` and `WithAdmin` contracts directly utilize solidity's `msg.sender` instead of relying on an OpenZeppelin `Context` contract. This prevents the contract admin from leveraging ERC2771 meta transactions, as they will have to be the direct `msg.sender` caller when calling admin methods `changeAdmin` and `setSuperOperator`.EIP2771

Recommendation: Use OpenZeppelin's `Context` contract and the inherited `_msgSender` method in the referenced contracts. This allows the referenced methods to receive ERC2771 meta transaction support in child contracts.
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Context.sol

Update: The Sandbox team states that "changes here would have distributed impact over many files, hence consciously not implementing it. We are simply not using meta transactions for admin calls."

QSP-8 Possible Inconsistency In `maxAllowedQuads`

Severity: Low Risk

Status: Mitigated

File(s) affected: `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`

Description: In the `setMaxAllowedQuads` function, the contract update the `maxAllowedQuads`value, which is used to limit the maximum allowed quads. The issue here is that the owner have this ability to change the `maxAllowedQuads` to any values including 0 which can block the `batchTransferQuadToL1` execution or change it to a value less than the older one which can cause inconsistency.

Recommendation:

- Consider verifying that `maxAllowedQuad` is greater than 0
- When `setMaxAllowedQuads` is called, verify that the new `maxAllowedQuads` is greater than the old value.

Update: The Sandbox team added a check to `setMaxAllowedQuads` that disallows a `_maxAllowedQuads` of 0.
`_maxAllowedQuads` can still be set to a value that is less than the current total number of quads which would result in an inconsistent contract state.

QSP-9 Renounceable Admin/Ownership

Severity: Informational

Status: Fixed

File(s) affected: `src/solc_0.8/common/BaseWithStorage/WithAdmin.sol`, `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`

Description:

- `WithAdmin`'s `changeAdmin` function does not perform any validation checks on `newAdmin`. This makes `admin` role renunciation possible, which would result in the forfeiting of access to all `admin` functionality.
- Same issue in the `PolygonLAndTunnel` contract using the Ownable library.

Recommendation:

- Remove `changeAdmin` and add `proposeAdmin`, which will propose a `newAdmin`, and `acceptAdmin`, which allows the `newAdmin` to accept the admin role.
- Override the `renounceOwernship` function to be reverted if it's called.

Update: The Sandbox team states that they "want ownership to be renounceable, hence are keeping it as it is."

QSP-10 Pragma Unlocked

Severity: Informational

Status: Fixed

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked". `ERC2771Handler` specifies a Solidity pragma version of `^0.8.0` which means that it can be compiled with any minor Solidity version release greater than or equal to `0.8.0`.

Recommendation: Change `ERC2771Handler` 's Solidity version to `0.8.2` to match the other audited contracts' Solidity version and lock it.

Update: The Sandbox team locked `ERC2771Handler`'s Solidity version at `0.8.2`.

QSP-11 Unused Interface

Severity: Informational

Status: Fixed

File(s) affected: `src/solc_0.8/common/interfaces/IPolygonLand.sol`, `src/solc_0.8/polygon/child/land/PolygonLandV1.sol`

Description: It is generally recommended that smart contract interfaces also be imported and implemented by the contracts which are intended to be targets of those interactions. This ensures that smart contracts' actual function definitions match the ones expected by smart contracts interacting with them.
The `IPolygonLand` interface is used by `PolygonLandTunnel` to interact with `PolygonLandV1`. However, `PolygonLandV1` doesn't inherit from the `IPolygonLand` interface.
Due to the interfaces aligning on the `PolygonLandV1` contract and the `IPolygonLand` interface, this issue has only been marked to "informational."

Recommendation: Modify `PolygonLandV1` to inherit from the `IPolygonLand` interface.

Update: The Sandbox team changed a parent contract of `PolygonLandV1`, `PolygonLandBaseToken`, to inherit from the `IPolygonLand` which will ensure that `PolygonLandV1` actually implements the interface that `PolygonLandTunnel` expects it to.

QSP-12 Incomplete Pauses

Severity: Undetermined

Status: Acknowledged

File(s) affected: `src/solc_0.8/polygon/root/land/LandTunnel.sol`, `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`

Description: The `owner` of `LandTunnel` can pause land transfers from Ethereum main net to Polygon but cannot pause transfers from Polygon to Ethereum main net.
Inversely, the `owner` of `PolygonLandTunnel` can pause land transfers from Polygon to Ethereum main net but cannot pause transfers from Ethereum main net to Polygon.
`LandTunnel` 's and `PolygonLandTunnel`'s partial pause functionality can lead to unintended behaviour when their respective `owner`'s call pause, but some of their functionality remains unpaused.

Recommendation: Modify `LandTunnel` and `PolygonLandTunnel` to pause Ethereum Mainnet to polygon transfers bilaterally.

Update: The Sandbox team states that "a single transaction which is executed on both L1 and L2, hence simultaneously pausing and resuming the tunnels, doesn't seem possible." However, the concern of the issue is not that a single transaction could pause both contracts at the same time. The issue is that separate owners can pause one contract but not the other, creating a one-way door into or out of Polygon.

QSP-13 Burned Quads Cannot Be Minted Again

Severity: Undetermined

Status: Fixed

File(s) affected: `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`, `src/solc_0.8/polygon/child/land/PolygonLandV1.sol`

Description: In the `_mintQuad` method it is checked whether the `_owners` slots are completely zeroed out. However besides storing the quad owner in the first 160 bits the `_owners` mapping also stores whether the quad token has been burnt (bit 161) and whether the token has an operator (bit 256). When quad tokens are burnt with any of the burn methods only the burn bit is set to `1`, the data is not zeroed out. Despite the token no longer being owned by anyone it cannot be minted via the `_mintQuad` method.
This may be intentional but the provided documentation does not provide any indication whether destroyed land tokens should be recoverable via the `_mintQuad` method.

Recommendation: Add documentation specifying whether Polygon Land tokens should be burnable. If they shouldn't, the burn and burnFrom methods should be disabled.

Update: The Sandbox team states that burnt quads not being able to be minted again "is a feature which sand had implemented, we won't be making any changes to resolve this as an issue."

Automated Analyses

Slither

The majority of the issues alerted by slither are false positive.

Code Documentation

- `src/solc_0.8/common/interfaces/IERC721MandatoryTokenReceiver.sol::L22` contains a commented out `supportsInterface` function. Either uncomment it if it is meant to be part of the `IERC721MandatoryTokenReceiver` interface, or remove it.
- `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol::L234` has a 337 column long comment. Break the comment into multiple lines to increase readability.
- `src/solc_0.8/common/interfaces/IPolygonLand.sol::L16` contains a commented out `exit` function. Either uncomment it if it is meant to be part of the `IPolygonLand` interface, or remove it.
- `src/solc_0.8/common/BaseWithStorage/ERC2771Handler.sol::L6` comments that `ERC2771Handler` is based off of OpenZepplin's implementation. Change the comment to reference a tagged version of the contract code that `ERC2771Handler` is based on to future-proof the comment from changes to OpenZepplin's master branch. Also, add comments listing what functionality was changed from OpenZepplin's implementation to increase readability.

Adherence to Best Practices

- Call `_ownerOf` from `_ownerAndOperatorEnabledOf` in `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol` instead of reimplementing ownership checking functionality in `_ownerAndOperatorEnabledOf` to increase readability and decrease deployment costs.
- Add indexing to `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`'s and `src/solc_0.8/polygon/root/land/LandTunnel.sol`'s events to make them easier for end-users and block explorers to filter for.
- `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`'s `childToken` and `src/solc_0.8/polygon/root/land/LandTunnel.sol`'s `rootToken` are only set in their constructors and should be made `immutable` to reduce gas costs.
- Functions visibility should be restricted as much as possible to minimize the chances of misuse.
  - Change `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`'s `pause` and `unpause` to have an `external` visibility instead of `public`.
  - Change `src/solc_0.8/polygon/root/land/LandTunnel.sol`'s `pause`, `unpause`, and `batchTransferQuadToL2` to have an `external` visibility instead of `public`.
- `src/solc_0.8/common/BaseWithStorage/WithAdmin.sol`: L27+28: Repeated retrieval of storage variable `_admin`. The `_admin` variable is read twice from storage, once on L27 when it's compared to the `msg.sender` and a second time when it's used an event parameter. It is recommended that unchanged storage variables that need to be reused multiple times within a method are stored in an intermediary local variable to save gas.
- Privileged checks prior to user-checks (`src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol` L41, L60, L163, L423; `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol` L249). It is checked whether the `msg.sender` is a "super operator" prior to whether the `msg.sender` is an "operator for all". It is recommended the "super operator" check be placed after the "operator for all" check in the OR-chain (`||`). This will improve gas usage for the average user while slightly increasing the gas cost of the methods for "super operators".
- `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol`: L435-451: Manual call data construction. While limiting the amount of gas passed for calls is generally not recommended as the gas cost of opcodes may be changed limiting gas can be done directly in solidity, without the use of assembly:

```
try
  IERC165(_contract).supportsInterface{gas: 10000}(interfaceID)
  returns (bool result)
{
  return result;
} catch {
  return false;
}
```

- `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol`: L323: Unnecessary check. Due to "super operators" having full access already, checking whether they are being approved or/not is unnecessary and adds an additional `SSTORE` operation for every call of `_setApprovalForAll`
- `src/solc_0.8/common/BaseWithStorage/ERC721BaseToken.sol`: L230-232: Defined constants underused. The `supportsInterface` method uses the inline defined literal function selector `0x01ffc9a7` and `0x80ac58cd` instead of using the previously defined `ERC165ID` constant
- `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`: L65-85: The `uint2str` method should be optimized and/or replaced with an existing implementation. It is recommended an existing implementation such as the one provided by OpenZeppelin be used (`contracts/utils/String.sol`, `Strings.toString`). The current implementation of `uint2str` could also be optimized (L79-81): `bstr[k] = bytes1(uint8(48 + _i % 10));`

- `src/solc_0.8/polygon/child/land/PolygonLandBaseToken.sol`: L141: Use of `require(false, revertString)` instead of `revert(revertString)`

1. Lack of indexed event fields (`src/solc_0.8/polygon/root/land/LandTunnel.sol`, `src/solc_0.8/polygon/child/land/PolygonLandTunnel.sol`, `src/solc_0.8/common/BaseWithStorage/WithSuperOperators.sol`, `src/solc_0.8/common/BaseWithStorage/WithAdmin.sol`). It is recommended that important event fields be indexed so that they can be queried more easily off-chain.

   - `src/solc_0.8/common/interfaces/ILandToken.sol`: Mismatched interface name. Besides not following the `I` interface naming convention the `LandToken` interface does not match the name of the file it is defined in `ILandToken.sol`. It is recommended that interfaces always be named with a leading 'I' and that their name matches the file they are defined in.

   - Reimplementation of access control logic (`src/solc_0.8/common/BaseWithStorage/WithSuperOperators.sol`, `src/solc_0.8/common/BaseWithStorage/WithAdmin.sol`). The referenced contracts implement access control patterns already available via common smart contract libraries. It is recommended that the `WithAdmin` contract be replaced with OpenZeppelin's `Ownable` and `WithSuperOperators` leverage OpenZeppelin's `AccessControl` or `AccessControlEnumerable` contracts.

Test Results

Test Suite Results

yarn test

```
  Asset:ERC1155
    bouncerAdmin
Nothing to compile
      √ can't set address 0 to bouncerAdmin (8086ms)
    mint
      √ minting an item results in a TransferSingle event (504ms)
    transfers
      √ transferring one instance of an item results in an ERC1155 TransferSingle event (550ms)
      √ transferring multiple instances of an item results in an ERC1155 TransferSingle event (549ms)
      √ transferring zero instances of an item results in an ERC1155 TransferSingle event (539ms)
      √ transferring an item with 1 supply does not result in an ERC1155 TransferBatch event (522ms)
      √ transferring an item with >1 supply does not result in an ERC1155 TransferBatch event (533ms)
      √ can be transferred to a normal address (520ms)
      √ cannot be transferred to zero address (454ms)
      √ cannot transfer more items than you own (446ms)
      √ cannot transfer an item with supply 1 that you do not own (453ms)
      √ cannot transfer an item that you do not own (448ms)
      √ cannot transfer more item of 1 supply (467ms)
      √ cannot transfer to a contract that does not accept ERC1155 (461ms)
      √ cannot transfer multiple instances of an item to a contract that does not accept ERC1155 (453ms)
      √ cannot transfer an item of supply 1 to a contract that does not accept ERC1155 (441ms)
      √ cannot transfer an item of supply 1 to a contract that does not return the correct ERC1155_IS_RECEIVER value (459ms)
      √ cannot transfer an item of supply >1 to a contract that does not return the correct ERC1155_IS_RECEIVER value (456ms)
    batch transfers
      √ transferring an item with 1 supply results in an ERC1155 BatchTransfer event (562ms)
      √ transferring an item with >1 supply results in an ERC1155 BatchTransfer event (552ms)
      √ transferring zero items with 1 supply results in an ERC1155 BatchTransfer event (522ms)
      √ transferring zero items with >1 supply results in an ERC1155 BatchTransfer event (538ms)
      √ transferring empty list results in an ERC1155 BatchTransfer event (550ms)
      √ transferring multiple items results in an ERC1155 BatchTransfer event (647ms)
      √ transferring multiple items including zero amount results in an ERC1155 BatchTransfer event (601ms)
      √ transferring an item with 1 supply with batch transfer does not result in a TransferSingle event (537ms)
      √ transferring an item with >1 supply with batch transfer does not result in a TransferSingle event (565ms)
      √ can use batch transfer to send tokens to a normal address (551ms)
      √ cannot batch transfer the same token twice and exceed the amount owned (470ms)
      √ can use batch transfer to send token twice if there is sufficient amount owned (591ms)
      √ cannot batch transfer tokens to zeroAddress (459ms)
      √ cannot batch transfer tokens if array lengths do not match (455ms)
      √ cannot batch transfer more than the amount owned (450ms)
      √ cannot batch transfer more items of 1 supply (451ms)
      √ cannot batch transfer to a contract that does not accept ERC1155 (466ms)
      √ cannot batch transfer to a contract that does not return the correct magic value (448ms)
      √ can batch transfer to a contract that does accept ERC1155 and which returns the correct magic value (609ms)
      √ can batch transfer item with 1 or more supply at the same time (569ms)
      √ can obtain balance of batch (568ms)
    approvalForAll
      √ setting approval results in ApprovalForAll event (501ms)
      √ setting approval fails if sender is operator (461ms)
      √ operator cannot transfer without approval (450ms)
      √ operator can transfer after approval (629ms)
      √ operator cannot transfer after approval is removed (682ms)
    supportsInterface
      √ contract claims to supports ERC165 (453ms)
      √ contract does not claim to support random interface (461ms)
      √ contract does not claim to support invalid interface (473ms)
    ordering
      √ transfer empty array (626ms)
      √ transfer multiple items in any order (i) (659ms)
      √ transfer multiple items in any order (ii) (675ms)
      √ transfer multiple items in any order (iii) (631ms)
      √ transfer multiple items in any order (iv) (744ms)
      √ transfer multiple items in any order (v) (647ms)
      √ transfer multiple items in any order (vi) (661ms)
      √ transfer multiple items in any order (vii) (656ms)
      √ transfer multiple items in any order (viii) (649ms)
      √ transfer multiple items in any order twice (i) (739ms)
      √ transfer multiple items in any order twice (ii) (883ms)
      √ transfer multiple items in any order twice (iii) (793ms)
      √ transfer multiple items in any order twice (iv) (837ms)
      √ transfer multiple items in any order twice (v) (801ms)
      √ transfer multiple items in any order twice (vi) (828ms)

  Asset:ERC721
    non existing NFT
      √ transfering a non existing NFT fails (184ms)
      √ tx balanceOf a zero owner fails (169ms)
      √ call balanceOf a zero owner fails (155ms)
      √ tx ownerOf a non existing NFT fails (169ms)
      √ call ownerOf a non existing NFT fails (171ms)
      √ tx getApproved a non existing NFT fails (140ms)
      √ call getApproved a non existing NFT fails (153ms)
    balance
      √ balance is zero for new user (168ms)
      √ balance return correct value (480ms)
    mint
      √ mint result in a transfer from 0 event (216ms)
      √ mint for gives correct owner (186ms)
    burnAsset
      √ burn result in a transfer to 0 event (275ms)
      √ burn result in ownerOf throwing (265ms)
    transfer
      √ transfering one NFT results in one erc721 transfer event (213ms)
      √ transfering one NFT change to correct owner (237ms)
      √ transfering one NFT increase new owner balance (267ms)
      √ transfering one NFT decrease past owner balance (276ms)
      √ transfering from without approval should fails (157ms)
      √ transfering to zero address should fails (153ms)
      √ transfering to a contract that do not accept erc721 token should not fail (263ms)
    safeTransfer
      √ safe transfering one NFT results in one erc721 transfer event (213ms)
      √ safe transfering to zero address should fails (156ms)
      √ safe transfering one NFT change to correct owner (248ms)
      √ safe transfering from without approval should fails (172ms)
      √ safe transfering to a contract that do not accept erc721 token should fail (191ms)
      √ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (202ms)
      √ safe transfering to a contract that do not implemented onERC721Received should fail (198ms)
      √ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (291ms)
    safeTransfer with empty bytes
      √ data:0x : safe transfering one NFT results in one erc721 transfer event (245ms)
      √ data:0x : safe transfering to zero address should fails (144ms)
      √ data:0x : safe transfering one NFT change to correct owner (244ms)
      √ data:0x : safe transfering from without approval should fails (170ms)
      √ data:0x : safe transfering to a contract that do not accept erc721 token should fail (204ms)
      √ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (201ms)
      √ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (172ms)
      √ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (292ms)
    safeTransfer with data
      √ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (245ms)
      √ data:0xff56fe3422 : safe transfering to zero address should fails (160ms)
      √ data:0xff56fe3422 : safe transfering one NFT change to correct owner (231ms)
      √ data:0xff56fe3422 : safe transfering from without approval should fails (156ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (218ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (202ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (187ms)
      √ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (307ms)
    ERC165
      √ claim to support erc165 (169ms)
      √ claim to support base erc721 interface (139ms)
      √ claim to support erc721 metadata interface (163ms)
      √ does not claim to support random interface (168ms)
      √ does not claim to support the invalid interface (153ms)
    Approval
      √ approving emit Approval event (186ms)
      √ removing approval emit Approval event (294ms)
      √ approving update the approval status (221ms)
      √ cant approve if not owner or operator  (237ms)
      √ approving allows transfer from the approved party (281ms)
      √ transfering the approved NFT results in aproval reset for it (334ms)
      √ transfering the approved NFT results in aproval reset for it but no approval event (323ms)
      √ transfering the approved NFT again will fail (333ms)
      √ approval by operator works (416ms)
    ApprovalForAll
      √ approving all emit ApprovalForAll event (213ms)
      √ approving all update the approval status (249ms)
      √ unsetting approval for all should update the approval status (327ms)
      √ unsetting approval for all should emit ApprovalForAll event (278ms)
      √ approving for all allows transfer from the approved party (332ms)
      √ transfering one NFT do not results in aprovalForAll reset (299ms)
      √ approval for all does not grant approval on a transfered NFT (327ms)
      √ approval for all set before will work on a transfered NFT (405ms)
      √ approval for all allow to set individual nft approve (501ms)

  GameToken:ERC721
    non existing NFT
      √ transfering a non existing NFT fails (3657ms)
      √ tx balanceOf a zero owner fails (293ms)
      √ call balanceOf a zero owner fails (251ms)
      √ tx ownerOf a non existing NFT fails (279ms)
      √ call ownerOf a non existing NFT fails (277ms)
      √ tx getApproved a non existing NFT fails (312ms)
      √ call getApproved a non existing NFT fails (295ms)
    balance
      √ balance is zero for new user (279ms)
      √ balance return correct value (575ms)
    mint
      √ mint result in a transfer from 0 event (408ms)
      √ mint for gives correct owner (347ms)
    burn
      √ burn result in a transfer to 0 event (464ms)
      √ burn result in ownerOf throwing (485ms)
    batchTransfer
      √ batch transfer of same NFT ids should fails (279ms)
      √ batch transfer works (388ms)
    mandatory batchTransfer
      √ batch transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (420ms)
      √ batch transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (348ms)
      √ batch transfering to a contract that do not accept erc721 token should fail (345ms)
      √ batch transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (344ms)
      √ batch transfering to a contract that do not implemented mandatory receiver should not fail (431ms)
      √ batch transfering to a contract that return the correct onERC721Received bytes shoudl succeed (424ms)
    mandatory transfer
      √ transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (379ms)
      √ transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (343ms)
      √ transfering to a contract that do not accept erc721 token should fail (342ms)
      √ transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (330ms)
      √ transfering to a contract that do not implemented mandatory receiver should not fail (402ms)
      √ transfering to a contract that return the correct onERC721Received bytes shoudl succeed (456ms)
    safe batch transfer
      √ safe batch transfer of same NFT ids should fails (295ms)
      √ safe batch transfer works (403ms)
    transfer
      √ transfering one NFT results in one erc721 transfer event (345ms)
      √ transfering one NFT change to correct owner (361ms)
      √ transfering one NFT increase new owner balance (376ms)
      √ transfering one NFT decrease past owner balance (388ms)
      √ transfering from without approval should fails (311ms)
      √ transfering to zero address should fails (296ms)
      √ transfering to a contract that do not accept erc721 token should not fail (383ms)
    safeTransfer
      √ safe transfering one NFT results in one erc721 transfer event (337ms)
      √ safe transfering to zero address should fails (281ms)
      √ safe transfering one NFT change to correct owner (374ms)
      √ safe transfering from without approval should fails (298ms)
      √ safe transfering to a contract that do not accept erc721 token should fail (329ms)
      √ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (345ms)
      √ safe transfering to a contract that do not implemented onERC721Received should fail (338ms)
      √ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (435ms)
    safeTransfer with empty bytes
      √ data:0x : safe transfering one NFT results in one erc721 transfer event (355ms)
      √ data:0x : safe transfering to zero address should fails (315ms)
      √ data:0x : safe transfering one NFT change to correct owner (324ms)
      √ data:0x : safe transfering from without approval should fails (312ms)
      √ data:0x : safe transfering to a contract that do not accept erc721 token should fail (329ms)
      √ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (344ms)
      √ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (307ms)
      √ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (435ms)
    safeTransfer with data
      √ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (338ms)
      √ data:0xff56fe3422 : safe transfering to zero address should fails (316ms)
      √ data:0xff56fe3422 : safe transfering one NFT change to correct owner (340ms)
      √ data:0xff56fe3422 : safe transfering from without approval should fails (330ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (314ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (347ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (343ms)
      √ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (435ms)
    ERC165
      √ claim to support erc165 (277ms)
      √ claim to support base erc721 interface (283ms)
      √ claim to support erc721 metadata interface (266ms)
      √ does not claim to support random interface (280ms)
      √ does not claim to support the invalid interface (266ms)
    Approval
      √ approving emit Approval event (349ms)
      √ removing approval emit Approval event (441ms)
      √ approving update the approval status (349ms)
      √ cant approve if not owner or operator  (392ms)
      √ approving allows transfer from the approved party (433ms)
      √ transfering the approved NFT results in aproval reset for it (453ms)
      √ transfering the approved NFT results in aproval reset for it but no approval event (434ms)
      √ transfering the approved NFT again will fail (408ms)
      √ approval by operator works (515ms)
    ApprovalForAll
      √ approving all emit ApprovalForAll event (323ms)
      √ approving all update the approval status (343ms)
      √ unsetting approval for all should update the approval status (424ms)
      √ unsetting approval for all should emit ApprovalForAll event (423ms)
      √ approving for all allows transfer from the approved party (456ms)
      √ transfering one NFT do not results in aprovalForAll reset (454ms)
      √ approval for all does not grant approval on a transfered NFT (463ms)
      √ approval for all set before will work on a transfered NFT (542ms)
      √ approval for all allow to set individual nft approve (593ms)

  SafeMathWithRequire
{ loopCounter: 81 }
{ loopCounter: 173 }
    √ cbrt6
{ loopCounter: 47 }
{ loopCounter: 139 }
    √ cbrt3
{ loopCounter: 215 }
{ loopCounter: 469 }
    √ rt6_3

  LandWeightedSANDRewardPool computation
    √ computing contributions (1174ms)

  MockSANDRewardPool
    √ Pool contains reward tokens (460ms)
    √ User with stakeTokens can stake (78ms)
    √ User earnings for 0 NFTs match expected reward (47ms)
    √ User earnings for 0 NFTs match expected reward with 1 stake (62ms)
    √ User earnings for 0 NFTs match expected reward with 2 stakes (64ms)
    √ User earnings for 0 NFTs match expected reward with 3 stakes (93ms)
    √ User earnings for 0 NFTs match expected reward with 4 stakes (92ms)
    √ User earnings for 0 NFTs match expected reward with 10 stakes (197ms)
    √ User earnings for 1 NFTs match expected reward (60ms)
    √ User earnings for 1 NFTs match expected reward with 10 stakes (204ms)
    √ User earnings for 2 NFTs match expected reward (77ms)
    √ User earnings for 3 NFTs match expected reward (94ms)
    √ User earnings for 3 NFTs match expected reward with 10 stakes (236ms)
    √ User earnings for 89 NFTs match expected reward (1447ms)
    √ User earnings for 89 NFTs match expected reward with 10 stakes (1718ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 2 users (90ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 2 users, 10 stakes each (359ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 3 users, 1 stake each (96ms)
    √ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 1 stake each (108ms)
    √ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 10 stakes each (423ms)
    √ Multiple Users' earnings for 3 NFTs match expected reward: 2 users, 1 stake each (204ms)
    √ Multiple Users' earnings for 100 NFTs match expected reward: 2 users, 1 stake each (3200ms)
    √ Staking with STAKE_AMOUNT plus an extra amount equivalent to 2 NFTs (61ms)
    √ Earlier staker gets more rewards with same NFT amount - small NFT number (92ms)
    √ Earlier staker gets more rewards with same NFT amount - large NFT number (3439ms)
    √ More lands give more rewards than earlier staker when NFT amounts are smaller (167ms)
    √ More lands do not give more rewards than earlier staker with large NFT amounts (3157ms)
    √ rewardToken in pool is more than amount notified (570ms)
    √ rewardToken in pool is zero (515ms)
    √ rewardToken in pool is less than amount notified (75ms)
    √ the call to notifyRewardAmount is made after users first call stake (521ms)
    √ user is earning rewards and pool is notified for a second time before end of current reward period (510ms)

  ActualSANDRewardPool
    √ Contract should exist (1492ms)
    √ Pool contains reward tokens
    √ User with stakeTokens can stake (65ms)
    √ User can earn rewardTokens if pool has been notified of reward (93ms)
    √ admin can notifyRewardAmount and start a new reward process (without sending more reward tokens) (47ms)
    √ User cannot earn rewardTokens if they stake after the end time (61ms)
    √ User earns full reward amount if they are the only staker after 1 day (63ms)
    √ User earns full reward amount if they are the only staker after 29 days (62ms)
    √ User with 0 LAND earns correct reward amount (104ms)
    √ User with 0 LAND earns correct reward amount - smaller stake (110ms)
    √ User with 1 LAND earns correct reward amount (154ms)
    √ User with 3 LANDs earns correct reward amount (172ms)
    √ User with 10 LANDs earns correct reward amount (267ms)
    √ User can withdraw some stakeTokens after several amounts have been staked (120ms)
    √ First user can withdraw their stakeTokens (107ms)
    √ User can withdraw all stakeTokens after several amounts have been staked (140ms)
    √ First user can claim their reward - no NFTs (140ms)
    √ First user can claim their reward - has NFTs (295ms)
    √ A user can claim their reward after multiple stakes (361ms)
    √ First user can exit the pool (119ms)
    √ A user can exit the pool after multiple stakes (158ms)
    √ A user with NFTs can exit the pool after multiple stakes (308ms)

  Catalyst_EPIC
    √ transfering from users[0] to users[1] should adjust their balance accordingly (1490ms)
    √ transfering from users[0] more token that it owns should fails (50ms)
    √ transfering to address zero should fails (62ms)
    √ transfering to address(this) should fail (46ms)
    √ transfering from users[0] to users[1] by users[0] should adjust their balance accordingly (107ms)
    √ transfering from users[0] by users[1] should fails (48ms)
    √ transfering from users[0] to users[1] should trigger a transfer event (74ms)
    √ transfering from users[0] to users[1] by operator after approval, should adjust their balance accordingly (187ms)
    √ transfering from users[0] to users[1] by operator after approval and approval reset, should fail (157ms)
    √ transfering from users[0] to users[1] by operator after approval, should adjust the operator alowance accordingly (155ms)
    √ transfering from users[0] to users[1] by operator after max approval (2**256-1), should NOT adjust the operator allowance (135ms)
    √ transfering from users[0] to users[1] by operator after approval, but without enough allowance, should fails (79ms)
    √ transfering from users[0] by operators without pre-approval should fails (62ms)
    √ approving operator should trigger a Approval event (73ms)
    √ disapproving operator (allowance to zero) should trigger a Approval event (141ms)
    √ approve to address zero should fails (50ms)

  Gem_POWER
    √ transfering from users[0] to users[1] should adjust their balance accordingly (1491ms)
    √ transfering from users[0] more token that it owns should fails (48ms)
    √ transfering to address zero should fails (45ms)
    √ transfering to address(this) should fail (61ms)
    √ transfering from users[0] to users[1] by users[0] should adjust their balance accordingly (105ms)
    √ transfering from users[0] by users[1] should fails (45ms)
    √ transfering from users[0] to users[1] should trigger a transfer event (76ms)
    √ transfering from users[0] to users[1] by operator after approval, should adjust their balance accordingly (188ms)
    √ transfering from users[0] to users[1] by operator after approval and approval reset, should fail (158ms)
    √ transfering from users[0] to users[1] by operator after approval, should adjust the operator alowance accordingly (154ms)
    √ transfering from users[0] to users[1] by operator after max approval (2**256-1), should NOT adjust the operator allowance (154ms)
    √ transfering from users[0] to users[1] by operator after approval, but without enough allowance, should fails (94ms)
    √ transfering from users[0] by operators without pre-approval should fails (62ms)
    √ approving operator should trigger a Approval event (75ms)
    √ disapproving operator (allowance to zero) should trigger a Approval event (140ms)
    √ approve to address zero should fails (49ms)

  LandBaseToken:ERC721
    non existing NFT
      √ transfering a non existing NFT fails (462ms)
      √ tx balanceOf a zero owner fails (68ms)
      √ call balanceOf a zero owner fails (78ms)
      √ tx ownerOf a non existing NFT fails (69ms)
      √ call ownerOf a non existing NFT fails (72ms)
      √ tx getApproved a non existing NFT fails (76ms)
      √ call getApproved a non existing NFT fails (78ms)
    balance
      √ balance is zero for new user (72ms)
      √ balance return correct value (291ms)
    mint
      √ mint result in a transfer from 0 event (89ms)
      √ mint for gives correct owner (96ms)
    burn
      √ burn result in a transfer to 0 event (167ms)
      √ burn result in ownerOf throwing (175ms)
    batchTransfer
      √ batch transfer of same NFT ids should fails (79ms)
      √ batch transfer works (148ms)
    mandatory batchTransfer
      √ batch transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (186ms)
      √ batch transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (88ms)
      √ batch transfering to a contract that do not accept erc721 token should fail (104ms)
      √ batch transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (109ms)
      √ batch transfering to a contract that do not implemented mandatory receiver should not fail (193ms)
      √ batch transfering to a contract that return the correct onERC721Received bytes shoudl succeed (159ms)
    mandatory transfer
      √ transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (167ms)
      √ transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (83ms)
      √ transfering to a contract that do not accept erc721 token should fail (107ms)
      √ transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (109ms)
      √ transfering to a contract that do not implemented mandatory receiver should not fail (181ms)
      √ transfering to a contract that return the correct onERC721Received bytes shoudl succeed (174ms)
    safe batch transfer
      √ safe batch transfer of same NFT ids should fails (79ms)
      √ safe batch transfer works (167ms)
    transfer
      √ transfering one NFT results in one erc721 transfer event (105ms)
      √ transfering one NFT change to correct owner (111ms)
      √ transfering one NFT increase new owner balance (140ms)
      √ transfering one NFT decrease past owner balance (138ms)
      √ transfering from without approval should fails (64ms)
      √ transfering to zero address should fails (79ms)
      √ transfering to a contract that do not accept erc721 token should not fail (202ms)
    safeTransfer
      √ safe transfering one NFT results in one erc721 transfer event (105ms)
      √ safe transfering to zero address should fails (79ms)
      √ safe transfering one NFT change to correct owner (137ms)
      √ safe transfering from without approval should fails (78ms)
      √ safe transfering to a contract that do not accept erc721 token should fail (112ms)
      √ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (90ms)
      √ safe transfering to a contract that do not implemented onERC721Received should fail (109ms)
      √ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (167ms)
    safeTransfer with empty bytes
      √ data:0x : safe transfering one NFT results in one erc721 transfer event (121ms)
      √ data:0x : safe transfering to zero address should fails (65ms)
      √ data:0x : safe transfering one NFT change to correct owner (136ms)
      √ data:0x : safe transfering from without approval should fails (78ms)
      √ data:0x : safe transfering to a contract that do not accept erc721 token should fail (112ms)
      √ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (79ms)
      √ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (109ms)
      √ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (168ms)
    safeTransfer with data
      √ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (107ms)
      √ data:0xff56fe3422 : safe transfering to zero address should fails (65ms)
      √ data:0xff56fe3422 : safe transfering one NFT change to correct owner (123ms)
      √ data:0xff56fe3422 : safe transfering from without approval should fails (63ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (97ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (106ms)
      √ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (108ms)
      √ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (168ms)
    ERC165
      √ claim to support erc165 (74ms)
      √ claim to support base erc721 interface (77ms)
      √ claim to support erc721 metadata interface (76ms)
      √ does not claim to support random interface (62ms)
      √ does not claim to support the invalid interface (78ms)
    Approval
      √ approving emit Approval event (92ms)
      √ removing approval emit Approval event (156ms)
      √ approving update the approval status (126ms)
      √ cant approve if not owner or operator  (125ms)
      √ approving allows transfer from the approved party (186ms)
      √ transfering the approved NFT results in aproval reset for it (189ms)
      √ transfering the approved NFT results in aproval reset for it but no approval event (155ms)
      √ transfering the approved NFT again will fail (175ms)
      √ approval by operator works (245ms)
    ApprovalForAll
      √ approving all emit ApprovalForAll event (108ms)
      √ approving all update the approval status (94ms)
      √ unsetting approval for all should update the approval status (157ms)
      √ unsetting approval for all should emit ApprovalForAll event (171ms)
      √ approving for all allows transfer from the approved party (125ms)
      √ transfering one NFT do not results in aprovalForAll reset (186ms)
      √ approval for all does not grant approval on a transfered NFT (155ms)
      √ approval for all set before will work on a transfered NFT (249ms)
      √ approval for all allow to set individual nft approve (313ms)

  Asset.sol
    √ user sending asset to itself keep the same balance (215ms)
    √ can transfer assets (94ms)
    √ user batch sending asset to itself keep the same balance (95ms)
    √ user batch sending in series whose total is more than its balance (90ms)
    √ user batch sending more asset that it owns should fails (40ms)
    √ can get the chainIndex from the tokenId
    √ can get the URI for an NFT (59ms)
    √ can get the URI for a FT (61ms)
    √ fails get the URI for an invalid tokeId
    √ can burn ERC1155 asset (91ms)
    √ can burn ERC721 asset (93ms)
    Asset: MetaTransactions
      √ can transfer by metaTx (120ms)
      √ fails to transfer someone else token by metaTx (108ms)
      √ can batch-transfer by metaTx (160ms)

  Asset_Giveaway
    √ User cannot claim when test contract holds zero assets (218ms)
    √ User can claim allocated multiple assets for multiple assetIds from Giveaway contract (368ms)
    √ Claimed Event is emitted for successful claim (59ms)
    √ User can claim allocated single asset for single assetId from Giveaway contract (47ms)
    √ User tries to claim the wrong amount of an assetID
    √ User cannot claim their assets more than once (59ms)
    √ User cannot claim assets from Giveaway contract if destination is not the reserved address
    √ User cannot claim assets from Giveaway contract to destination zeroAddress
    √ User cannot claim assets from Giveaway contract with incorrect asset param
    √ User can claim allocated multiple assets for multiple assetIds from alternate address (412ms)
    √ merkleRoot cannot be set twice (173ms)
    √ merkleRoot can only be set by admin

  GAS:Asset_Giveaway_1:Claiming
    √ 1 claim (268ms)
    √ 10 claims (287ms)
    √ 4000 claims (1085ms)
    √ 10000 claims (2563ms)
{
  "Gas per claim - 1 claim total": 112331,
  "Gas per claim - 10 claims total": 115008,
  "Gas per claim - 4000 claims total": 122100,
  "Gas per claim - 10000 claims total": 123886
}

  MerkleTree_assets
    √ should validate the data

  SignedGiveaway.sol
    initialization
      √ interfaces (780ms)
    roles
      √ admin
      √ signer
    claim
      √ should be able to claim sand (106ms)
      √ should fail to claim the same id twice (69ms)
      √ should fail to claim if the signature is wrong (50ms)
      √ should fail to mint if the signer is invalid (44ms)
      √ claim with metaTX trusted forwarder (133ms)
    revoke
      √ should fail to revoke if not admin
      √ should fail to claim if the id was revoked (59ms)
    pause
      √ should fail to pause if not admin
      √ should fail to unpause if not admin
      √ should fail to claim if paused (62ms)
      √ should be able to claim sand after pause/unpause (149ms)
    coverage
      √ a valid signature must verify correctly (49ms)
      √ check the domain separator

  SafeMathWithRequire.sol library via MockSafeMathWithRequire.sol
    √ sqrt6, sqrt multiplied by 1e6
    √ sqrt3, sqrt multiplied by 1e3
    √ cbrt6, cube root multiplied by 1e6 (46ms)
    √ cbrt3, cube root multiplied by 1e3

  LandContributionCalculator
    roles
      √ admin should be able to call setNFTMultiplierToken (486ms)
      √ others should fail to call setNFTMultiplierToken (157ms)
    calculation
      √ zero lands
      √ 1 lands (64ms)
      √ 2 lands (97ms)
      √ 3 lands (109ms)
      √ 4 lands (124ms)
      √ 5 lands, to high to be minted (82ms)
      √ 10 lands, to high to be minted (91ms)
      √ 20 lands, to high to be minted (96ms)
      √ 50 lands, to high to be minted (84ms)
      √ 100 lands, to high to be minted (89ms)
      √ 408 lands, to high to be minted (94ms)
      √ 166464 lands, to high to be minted (92ms)
      √ 67917312 lands, to high to be minted (90ms)

  LandOwnerContributionCalculator
    roles
      √ admin should be able to call setNFTMultiplierToken (516ms)
      √ others should fail to call setNFTMultiplierToken (171ms)
    calculation
      √ users without lands get zero contributions (49ms)
      √ 1 lands (64ms)
      √ 2 lands (60ms)
      √ 3 lands (93ms)
      √ 4 lands (94ms)

  PeriodicRewardCalculator
    √ only Admin can call setDuration (213ms)
    √ calling setDuration should update campaign duration (47ms)
    √ calling setDuration during the campaing should fail
    roles
      √ reward pool should be able to call restartRewards
      √ others should fail to call restartRewards
      √ reward distribution should be able to call notifyRewardAmount
      √ other should fail to call notifyRewardAmount
      √ reward distribution should be able to call setSavedRewards
      √ other should fail to call setSavedRewards
    should be no rewards on initialization
      √ startup
      √ restart call (110ms)
    reward distribution
      √ setup: we use the rate, so REWARDS must be multiple of duration (or leftover + reward if we add in the middle) (117ms)
      √ we distribute rewards linearly (1368ms)
      √ if restart is called (with contribution!=0) then rewards starts from zero again (1025ms)
      √ calling notifyRewardAmount in the middle of the distribution will distribute the remaining + what was added (1432ms)
      √ calling notifyRewardAmount after the distribution will distribute both amounts (1386ms)

  TwoPeriodsRewardCalculator
    √ startup (322ms)
    roles
      √ reward pool should be able to call restartRewards
      √ others should fail to call restartRewards
      √ reward distribution should be able to call (c) => c.runCampaign(12345678, 9876)
      √ other should fail to call (c) => c.runCampaign(12345678, 9876)
      √ reward distribution should be able to call (c) => c.setInitialCampaign(12345678, 9876)
      √ other should fail to call (c) => c.setInitialCampaign(12345678, 9876)
      √ reward distribution should be able to call (c) => __awaiter(this, void 0, void 0, function* () {
            yield c.setInitialCampaign(123, 1234);
            return c.updateNextCampaign(12345678, 9876);
        })
      √ other should fail to call (c) => __awaiter(this, void 0, void 0, function* () {
            yield c.setInitialCampaign(123, 1234);
            return c.updateNextCampaign(12345678, 9876);
        })
    setup restrictions
      √ should fail to set initial campaign if campaign is running (54ms)
      √ should fail to set next campaign if no campaign is running
      √ should fail to update current campaign if no campaign is running
      √ run campaign always works (258ms)
    reward distribution
      √ run an initial campaign alone (200ms)
      √ run initial and next campaign (359ms)
      √ run next campaign after the initial one finished (562ms)
    restart reward
      √ before everything (106ms)
      √ in middle of the first campaign (78ms)
      √ in middle of the second campaign (93ms)
      √ after everything (95ms)
      √ intermixed (168ms)

  SandRewardPool
    √ last time reward application should match the duration (1848ms)
    √ total supply is at first empty
    √ staking should update the reward balance, supply and staking token balance (140ms)
    √ withdraw should update the reward balance, supply and staking token (155ms)
    √ reward per token should be 0 if total supply is 0
    √ reward per token calculation (122ms)
    √ earned calculation (139ms)
    √ get reward should transfer the reward and emit an event (171ms)
    √ exiting should withdraw and transfer the reward (142ms)
    √ pool contains reward tokens (46ms)
    √ user can earn reward tokens if pool has been notified of reward (124ms)
    √ admin can notify to start a new reward process (without sending more reward tokens) (103ms)
    √ user cannot earn rewardTokens if they stake after the end time (64ms)
    √ user earns full reward amount if there is only one staker after 1 day(s) (109ms)
    √ user earns full reward amount if there is only one staker after 27 day(s) (89ms)
    √ User with 0 LAND earns correct reward amount (139ms)
    √ User with 0 LAND earns correct reward amount - smaller stake (123ms)
    √ User with 1 LAND(s) earns correct reward amount (187ms)
    √ User with 3 LAND(s) earns correct reward amount (197ms)
    √ User with 10 LAND(s) earns correct reward amount (317ms)
    √ User can withdraw some stakeTokens after several amounts have been staked (105ms)
    √ First user can claim their reward - no NFTs (172ms)
    √ First user can claim their reward - has NFTs (357ms)
    √ A user can claim their reward after multiple stakes (450ms)
    √ First user can exit the pool (154ms)
    √ A user can exit the pool after multiple stakes (207ms)
    √ A user with NFTs can exit the pool after multiple stakes (436ms)
    √ Change externals contracts
    √ user earnings for 0 NFT(s) match expected reward with 1 stake(s) (111ms)
    √ user earnings for 0 NFT(s) match expected reward with 2 stake(s) (155ms)
    √ user earnings for 0 NFT(s) match expected reward with 4 stake(s) (171ms)
    √ user earnings for 0 NFT(s) match expected reward with 10 stake(s) (343ms)
    √ user earnings for 1 NFT(s) match expected reward with 1 stake(s) (125ms)
    √ user earnings for 1 NFT(s) match expected reward with 10 stake(s) (464ms)
    √ user earnings for 2 NFT(s) match expected reward with 1 stake(s) (154ms)
    √ user earnings for 3 NFT(s) match expected reward with 1 stake(s) (168ms)
    √ user earnings for 3 NFT(s) match expected reward with 10 stake(s) (597ms)
    √ user earnings for 89 NFT(s) match expected reward with 1 stake(s) (1513ms)
    √ user earnings for 89 NFT(s) match expected reward with 10 stake(s) (1961ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 2 users, 10 stake each (643ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 3 users, 1 stake each (179ms)
    √ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 1 stake each (192ms)
    √ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 10 stake each (189ms)
    √ Multiple Users' earnings for 3 NFTs match expected reward: 2 users, 1 stake each (283ms)
    √ Multiple Users' earnings for 100 NFTs match expected reward: 2 users, 1 stake each (3343ms)
    √ Staking with STAKE_AMOUNT plus an extra amount equivalent to 2 NFTs (137ms)
    √ Earlier staker gets more rewards with same NFT amount - small NFT number (172ms)
    √ Earlier staker gets more rewards with same NFT amount - large NFT number (3331ms)
    √ More lands give more rewards than earlier staker when NFT amounts are smaller (208ms)
    √ More lands do not give more rewards than earlier staker with large NFT amounts (3413ms)
    √ rewardToken in pool is more than amount notified (108ms)
    √ rewardToken in pool is zero (117ms)
    √ rewardToken in pool is less than amount notified (124ms)
    √ the call to notifyRewardAmount is made after users first call stake (99ms)
    √ user is earning rewards and pool is notified for a second time before end of current reward period (130ms)
    √ Multiplier & reward are correct (310ms)
    - THIS IS FALSE, EVERYBODY CAN DO IT: Only sender or reward distribution can compute sender's account

  LandOwnersSandRewardPool
    √ users with land should be able to stake (2065ms)
    √ users without land should revert
    √ if a user sells his land we can recompute the contribution (573ms)

  new SandRewardPool main contract tests
    roles
      √ admin should be able to call setContributionCalculator (934ms)
      √ other should fail to call setContributionCalculator (56ms)
      √ admin should be able to call setRewardToken
      √ other should fail to call setRewardToken (51ms)
      √ admin should be able to call setStakeToken
      √ other should fail to call setStakeToken (38ms)
      √ admin should be able to call setRewardCalculator
      √ other should fail to call setRewardCalculator (53ms)
      √ admin should be able to call recoverFunds (121ms)
      √ other should fail to call recoverFunds (50ms)
      √ recoverFunds must fail with address zero
    reward distribution
      only one user
        √ reward before stake (364ms)
        √ stake before rewards (348ms)
        √ stake->withdraw so total contribution == 0, stake again (296ms)
        √ stake->withdraw so total contribution == 0, stake again with some rewards (432ms)
      two users
        √ reward before stake (375ms)
        √ user1 stake before rewards (356ms)
        √ user2 stake before rewards (408ms)
      10 users
        √ reward before stake (1233ms)
    contribution calculation
      √ initial (468ms)
      √ computeContribution after the user change his contribution (903ms)
      √ computeContributionInBatch after the user change his contribution (779ms)
    contribution calculation with rewards
      √ initial (1291ms)
      √ computeContribution after users change his contribution (2048ms)
      √ computeContributionInBatch after the user change his contribution (1240ms)
    trusted forwarder and meta-tx
      √ should fail to set the trusted forwarder if not admin (54ms)
      √ should success to set the trusted forwarder if admin (47ms)
      √ setReward with meta-tx (120ms)
      √ stake with meta-tx (141ms)
      √ withdraw with meta-tx (149ms)

  new SandRewardPool anti compound tests
    √ user can only get his rewards after lockTimeMS (323ms)
    √ we can disable lockTimeMS check by setting it to zero (151ms)
    roles
      √ admin should be able to call setAntiCompoundLockPeriod (63ms)
      √ other should fail to call setAntiCompoundLockPeriod (66ms)

  ERC677Token
    √ Transfering tokens to ERC677Receiver contract should emit an OnTokenTransferEvent event (1917ms)
    √ Transfering tokens to EOA (93ms)
    √ Transfering tokens to a non receiver contract should fail
    √ Transfering tokens to a contract with fallback function should succeed (41ms)

  use withSnapshot to keep your testing environment clean
    √ withSnapshot doesn't care about what happen before (588ms)

  Faucet
    √ Send cannot exceed Faucet limit amout (671ms)
    √ Send cannot be executed twice without awaiting (112ms)
    √ Send succeded for correct asked amount (123ms)
    √ Retrieve succeded for deployer (111ms)
    √ Retrieve succeded for deployer with any address (92ms)
    √ Retrieve fail for user that is not deployer (48ms)
    √ setPeriod succeed for deployer (60ms)
    √ setPeriod fail for user that is not deployer
    √ setLimit succeed for deployer (68ms)
    √ Send with new limit succeed after limit update. (190ms)
    √ setLimit fail for user that is not deployer

  GameMinter
    GameMinter: Calling Directly
      √ should fail to create GAME if user has insufficient SAND
      √ should allow anyone to create a game (102ms)
      √ should allow owner to add assets (95ms)
      √ should charge a fee when owner adds assets (75ms)
      √ should allow editor to add assets (127ms)
      √ should allow owner to remove assets (125ms)
      √ should allow editor to remove assets (92ms)
      √ should fail if not authorized to add assets
      √ should fail to modify GAME if user has insufficient SAND
      √ should fail if not authorized to remove assets
      √ should fail if not authorized to set GAME URI
      √ allows GAME owner to set GAME URI (110ms)
      √ allows GAME editor to set GAME URI (108ms)
    GameMinter: Sandbox MetaTXs
      √ should allow anyone to create a game via MetaTx (173ms)
      √ should allow GAME Owner to add assets via MetaTx (156ms)
      √ should allow GAME Owner to remove assets via MetaTx (139ms)
      √ should allow GAME Owner to set URI via MetaTx (167ms)
      √ should allow GAME Editor to add assets via MetaTx (114ms)
      √ should allow GAME Editor to remove assets via MetaTx (78ms)
      √ should allow GAME Editor to set URI via MetaTx (120ms)

  GameToken
    GameToken: Minting GAMEs
      √ can update the GameMinter address (46ms)
      √ Minter can create GAMEs when _Minter is set (93ms)
      √ should revert if trying to reuse a baseId
      √ gameId contains creator, randomId, chainIndex & version data
      √ can get the storageId for a GAME
      √ can get the chainIndex for a GAME
```

```
      √ reverts if non-minter trys to mint Game when _Minter is set
      GameToken: Mint With Assets
        √ fails to create if "to" address is the gameToken contract
        √ fails to add ERC1155 tokens to the game if Operator != GAME contract (70ms)
        √ fails to add ERC1155 token batch to the game if Operator != GAME contract (128ms)
        √ can mint Games with single Asset (171ms)
        √ can mint Games with many Assets (265ms)
        √ should fail if length of assetIds and values dont match (55ms)
      GameToken: Modifying GAMEs
        √ should allow the owner to add game editors (59ms)
        √ should allow the owner to remove game editors (45ms)
        √ should revert if non-owner trys to set Game Editors
        √ Minter can add single Asset (276ms)
        √ should bump the version number in the gameId
        √ Minter can add multiple Assets (296ms)
        √ Minter can remove single Asset (139ms)
        √ fails when removing more assets than the game contains
        √ Minter can remove multiple Assets (176ms)
        √ Game token should acurately track token balances for owners (59ms)
    GameToken: Transferring GAMEs
      √ current owner can transfer ownership of a GAME (63ms)
      √ can transfer creatorship of a GAME
      √ transfer creatorship should revert for a non existing game
      √ can transfer creatorship of a GAME back to original creator (70ms)
      √ should fail if non-owner trys to transfer a GAME
      √ transfer creatorship should revert for a burned game
    GameToken: MetaData
      √ can get the ERC721 token contract name
      √ can get the ERC721 token contract symbol
      √ can get the tokenURI
      √ Minter can set the tokenURI (64ms)
      √ should revert if ownerOf == address(0)
      √ should revert if not Minter
      √ should be able to retrieve the creator address from the gameId
    GameToken: Destroying Games
      √ fails if "from" != game owner
      √ fails if sender != game owner and not metatx
      GameToken: burnAndRecover
        √ fails if "to" == address(0)
        √ fails to destroy if "to" == Game Token contract
        √ fails if "from" != game owner
        √ fails if sender != game owner and not metatx
        √ can destroy GAME and recover assets in 1 tx if not too many assets (146ms)
        √ creatorOf() should should still return original creator
        √ game should no longer exist
      GameToken: Destroy... then Recover
        √ fails to recover if the GAME token has not been burnt
        √ can destroy without transfer of assets (116ms)
        √ fails to recover if "to" address is the gameToken contract
        √ fails to recover assets if caller is not from or validMetaTx
        √ can recover remaining assets from burnt GAME in batches (177ms)
    GameToken: Token Immutability
      √ should store the creator address, subID & version in the gameId
      √ should consider future versions of gameIds as invalid
      √ should update version when changes are made (61ms)
      √ should use baseId (creator address + subId) to map to game Assets
    GameToken: MetaTransactions
      √ can get isTrustedForwarder
      √ can call setGameEditor via metaTx (65ms)
      √ can call burnFrom via metaTx (235ms)
      √ can call recoverAssets via metaTx (185ms)
      √ can call transferCreatorship via metaTx (57ms)

  AuthValidator
    √ signature should be valid (490ms)
    √ signature should be invalid

  EstateSaleWithAuth
    √ should be able to purchase with valid signature (3737ms)
    √ should NOT be able to purchase with invalid signature
    √ should be able to purchase through sand contract (67ms)

  GAS:Multi_Giveaway_1:Claiming
    √ 1 claim (5430ms)
    √ 10 claims (1178ms)
    √ 4000 claims (5501ms)
    √ 10000 claims (11477ms)
{
  "Gas per claim - 1 claim total": 200603,
  "Gas per claim - 10 claims total": 203263,
  "Gas per claim - 4000 claims total": 210403,
  "Gas per claim - 10000 claims total": 212162
}

  MerkleTree_multi
    √ should validate the data

  Multi_Giveaway
    Multi_Giveaway_common_functionality
      √ Admin has the correct role (724ms)
      √ Admin can add a new giveaway (44ms)
      √ Cannot add a new giveaway if not admin
      √ User can get their claimed status (733ms)
      √ Claimed status is correctly updated after allocated tokens are claimed - 2 claims of 2 claimed (1387ms)
      √ Claimed status is correctly updated after allocated tokens are claimed - 1 claim of 2 claimed (77ms)
      √ MultiGiveaway contract returns ERC721 received (755ms)
      √ MultiGiveaway contract returns ERC721 Batch received
      √ MultiGiveaway contract returns ERC1155 received for supply 1
      √ MultiGiveaway contract returns ERC1155 received
      √ MultiGiveaway contract returns ERC1155 Batch received
    Multi_Giveaway_single_giveaway
      √ User cannot claim when test contract holds no tokens
      √ User cannot claim sand when contract does not hold any (1135ms)
      √ User can claim allocated multiple tokens from Giveaway contract (1441ms)
Number of assets: 64 ; Gas used: 2048544
      √ User can claim allocated 64 tokens from Giveaway contract (3142ms)
      √ Claimed Event is emitted for successful claim (1183ms)
      √ User can claim allocated ERC20 from Giveaway contract when there are no assets or lands allocated (75ms)
      √ User cannot claim if they claim the wrong amount of ERC20
      √ User cannot claim more than once (76ms)
      √ User cannot claim from Giveaway contract if destination is not the reserved address
      √ User cannot claim from Giveaway contract to destination zeroAddress
      √ User cannot claim from Giveaway contract to destination MultiGiveaway contract address
      √ User cannot claim from Giveaway if ERC1155 contract address is zeroAddress (1131ms)
      √ User cannot claim from Giveaway if ERC721 contract address is zeroAddress
      √ User cannot claim from Giveaway if ERC20 contract address is zeroAddress
      √ User cannot claim from Giveaway if ERC20 contract address array length does not match amounts array length
      √ User cannot claim from Giveaway if ERC1155 values array length does not match ids array length
      √ User cannot claim after the expiryTime (1178ms)
      √ User cannot claim if expiryTime is 0 (49ms)
    Multi_Giveaway_two_giveaways
      √ User cannot claim when test contract holds no tokens - multiple giveaways, 1 claim (745ms)
      √ User cannot claim sand when contract does not hold any - multiple giveaways, 1 claim (1287ms)
      √ User can claim allocated multiple tokens from Giveaway contract - multiple giveaways, 1 claim (1618ms)
      √ User can claim allocated multiple tokens from Giveaway contract - multiple giveaways, 2 claims (670ms)
      √ User cannot claim from Giveaway contract if the claims array length does not match merkle root array length
      √ User cannot claim from Giveaway contract if the claims array length does not match proofs array length
      √ User cannot claim allocated tokens from Giveaway contract more than once - multiple giveaways, 2 claims (120ms)
    Multi_Giveaway_single_claim
      √ User cannot claim when test contract holds no tokens (702ms)
      √ User cannot claim sand when contract does not hold any (1088ms)
      √ User can claim allocated multiple tokens from Giveaway contract (1462ms)
      √ Claimed Event is emitted for successful claim (62ms)
      √ User cannot claim more than once (73ms)
    Trusted_forwarder_and_meta-tx
      √ should fail to set the trusted forwarder if not admin (699ms)
      √ should succeed in setting the trusted forwarder if admin
      √ claim with meta-tx: user can claim from single giveaway using single claim function (1465ms)
      √ claim with meta-tx: user cannot claim from single giveaway using single claim function more than once (173ms)
      √ claim with meta-tx: user can claim from single giveaway using multiple claim function (394ms)
      √ claim with meta-tx: user cannot claim from single giveaway using multiple claim function more than once (139ms)
      √ claim with meta-tx: user can claim from multiple giveaways (1987ms)
      √ claim with meta-tx: user cannot claim from multiple giveaways more than once (195ms)

  Permit
    √ ERC20 Approval event is emitted when msg signer == owner (643ms)
    √ Nonce is incremented for each Approval (62ms)
    √ Permit function reverts if deadline has passed
    √ Permit function reverts if owner is zeroAddress
    √ Permit function reverts if owner != msg signer
    √ Permit function reverts if spender is not the approved spender
    √ Domain separator is public
    √ Non-approved operators cannot transfer ERC20 until approved (138ms)
    √ Approved operators cannot transfer more ERC20 than their allowance (69ms)
    √ Approved operators cannot transfer more ERC20 than there is (74ms)

  PolygonAsset.sol
    √ user sending asset to itself keep the same balance (5746ms)
    √ user batch sending asset to itself keep the same balance (89ms)
    √ user batch sending in series whose total is more than its balance (62ms)
    √ user batch sending more asset that it owns should fails (54ms)
    √ can get the chainIndex from the tokenId
    √ can get the URI for an NFT (59ms)
    √ can get the URI for a FT (59ms)
    √ fails get the URI for an invalid tokeId
    √ can burn ERC1155 asset (76ms)
    √ can burn ERC721 asset (76ms)
    PolygonAsset: MetaTransactions
      √ can transfer by metaTx (108ms)
      √ fails to transfer someone else token by metaTx (124ms)
      √ can batch-transfer by metaTx (155ms)
    Asset <> PolygonAsset: Transfer
      √ can transfer L1 minted assets: L1 to L2 (9076ms)
      √ can transfer L2 minted assets: L2 to L1 (217ms)
      √ can transfer multiple L1 minted assets: L1 to L2 (399ms)
      √ can transfer partial supplies of L1 minted assets: L1 to L2 (503ms)
      √ can transfer multiple L2 minted assets: L2 to L1 (419ms)
      √ can transfer partial supplies of L2 minted assets: L2 to L1 (597ms)
      √ can transfer assets from multiple L1 minted batches: L1 to L2 (603ms)
      √ can transfer assets from multiple L2 minted batches: L2 to L1 (642ms)
      √ can return L1 minted assets: L1 to L2 to L1 (377ms)
      √ can return L2 minted assets: L2 to L1 to L2 (321ms)
      √ Deposit 1 asset from 20 ERC1155 L1 to L2 (207ms)
      Transfer Gems and catalyst L1 to L2
        √ Deposit asset from L1 to L2 with 1 catalyst legendary and 4 power gems (296ms)
        √ Deposit asset from L1 to L2 with 1 catalyst legendary (259ms)
        √ Deposit asset from L1 to L2 with 1 catalyst legendary 1 gem defense (236ms)
        √ Deposit asset from L1 to L2 with catalyst or gems out of bound  (118ms)
        √ Deposit asset from L1 to L2 without catalyst and gems (266ms)
      Transfer Gems and catalyst L2 to L1
        √ Deposit asset from L2 to L1 with 1 catalyst legendary and 4 power gems (235ms)
        √ Deposit asset from L2 to L1 with 1 catalyst legendary (230ms)
        √ Deposit asset from L2 to L1 with 1 catalyst legendary 1 gem defense (238ms)
        √ Deposit asset from L2 to L1 without catalyst and gems (216ms)

  assetSignedAuctionAuth
    √ should be able to claim seller offer in ETH (169ms)
    √ should NOT be able to claim offer if signature mismatches (54ms)
    √ should be able to claim seller offer in SAND (164ms)
    √ should be able to claim seller offer with basic signature (159ms)
    √ should be able to cancel offer (59ms)
    √ should NOT be able to claim offer without sending ETH (54ms)
    √ should NOT be able to claim offer without enough SAND (113ms)
    √ should NOT be able to claim offer if it did not start yet (53ms)
    √ should NOT be able to claim offer if it already ended (62ms)

  PolygonBundleSandSale.sol
    √ should fail to deploy with a zero receiving wallet (258ms)
    √ assuming that the medianizer return the price in u$s * 1e18 and usdAmount is also in u$s * 1e18. There is no need to round up at most you loose 1e-18 u$s. (70ms)
    √ onERC1155Received should fail if called directly
    setReceivingWallet
      √ should fail to setReceivingWallet if not admin
      √ should fail if address is zero
      √ admin should success to setReceivingWallet
    create Sale
      √ NFT can't be used with numPacks > 1 ? (86ms)
      √ NFT can be used with numPacks == 1  (145ms)
      √ single sale (139ms)
      √ multiple/batch sale (169ms)
    Withdraw
      √ withdraw (434ms)
      √ should fail to withdraw if not admin (159ms)
    Buy packs with DAI
      √ should fail with the wrong saleId
      √ should fail if not enough packs (226ms)
      √ should fail if not enough DAI (287ms)
      √ buy with DAI, obs: buyer != to (513ms)
      Buy packs with ETH
        √ should fail with the wrong saleId
        √ should fail if not enough packs (205ms)
        √ should fail if not enough ETH (236ms)
        √ buy with ETH, obs: buyer != to (535ms)

  AssetAttributesRegistry
    √ getRecord for non existing assetId (3889ms)
    √ setCatalyst for legendary catalyst with 4 gems (200ms)
    √ setCatalyst should fail for non minter account
    √ setCatalyst with gems.length > MAX_NUM_GEMS should fail (54ms)
    √ setCatalyst with gems.length > maxGemForCatalyst should fail (43ms)
    √ setCatalystWithBlockNumber should fail for non migration contract
    √ addGems to rareCatalystId (219ms)
    √ should fail for non-nft
    √ addGems should fail for non minter account (163ms)
    √ addGems should fail for empty gemsId array
    √ addGems should fail for non existing catalystId
    √ should fail for gemId = 0
    √ addGems should fail when trying to add two gems in total to commonCatalyst (194ms)
    √ admin can change attributes contract
    √ fails if anyone other than admin trys to change attributes

  AssetAttributesRegistry: getAttributes
    getAttributes: minting
      √ can get attributes for 1 gem (8127ms)
      √ can get attributes for 2 identical gems (224ms)
      √ can get attributes for 3 identical gems (234ms)
      √ can get attributes for 4 identical gems (233ms)
      √ can get attributes for 2 different gems (223ms)
      √ can get attributes for 3 different gems (239ms)
      √ can get attributes for 4 different gems (234ms)
      √ can get attributes for 2 identical gems + 1 different gem (229ms)
      √ can get attributes for 3 identical gems + 1 different gem (230ms)
      √ can get attributes for 2 identical gems + 2 different identical gems (226ms)
    getAttributes: upgrading
      √ can get attributes when adding 1 gem to an asset with an empty catalyst (289ms)
      √ can get attributes when adding 2 identical gems to an asset with an empty catalyst (309ms)
      √ can get attributes when adding 3 identical gems to an asset with an empty catalyst (297ms)
      √ can get attributes when adding 4 identical gems to an asset with an empty catalyst (328ms)
      √ can get attributes when adding 2 different gems to an asset with an empty catalyst (292ms)
      √ can get attributes when adding 3 different gems to an asset with an empty catalyst (282ms)
      √ can get attributes when adding 4 different gems to an asset with an empty catalyst (312ms)
      √ can get attributes when adding 1 similar gem to an asset with existing gems (311ms)
      √ can get attributes when adding 1 different gem to an asset with existing gems (280ms)
      √ can get attributes when adding 2 similar gems to an asset with existing gems (296ms)
      √ can get attributes when adding 2 different gems to an asset with existing gems (314ms)
      √ can get attributes when adding 3 similar gems to an asset with existing gems (313ms)
      √ can get attributes when adding 3 different gems to an asset with existing gems (302ms)
      √ can get attributes when adding gems to an asset multiple times (344ms)
      √ can get attributes when upgrading an asset multiple times (711ms)
      √ attributes after multiple upgrades are correct (321ms)
      √ should fail if numGems > MAX-NUM_GEMS (371ms)

  AssetMinter
    AssetMinter: Mint
      √ the assetMInterAdmin is set correctly (5019ms)
      √ the assetMinter quantities are set correctly (283ms)
      √ Record is created with correct data on minting with legendary catalyst (NFT) (8088ms)
      √ Transfer event is emitted on minting an NFT (catalyst legendary) (889ms)
      √ CatalystApplied event is emitted on minting an NFT with a catalyst (1184ms)
      √ Catalysts and gems totalSuplies are reduced when added (1230ms)
      √ Mint without catalyst (1112ms)
      √ Mint custom number admin (126ms)
      √ Mint custom number user3 (125ms)
      AssetMinter: MintMultiple
        √ TransferBatch event is emitted on minting a single FT via mintMultiple (968ms)
        √ TransferBatch event is emitted on minting a multiple FTs (489ms)
        √ CatalystApplied event is emitted for each NFT minted with a catalyst (1537ms)
        √ records should be updated correctly for each asset minted (356ms)
        √ totalSupply & balance should be reduced for burnt gems & catalysts (1268ms)
      AssetMinter: addGems
        √ Can extract an erc721 & add Gems (8064ms)
      AssetMinter: Failures
        √ should fail if "to" == address(0)
        √ should fail if "from" != _msgSender()
        √ should fail if gem == Gem(0)
        √ should fail if gemIds.length > MAX_NUM_GEMS (127ms)
        √ should fail if gemIds.length > maxGems (127ms)
        √ minting WO catalyst: should fail if asstID = 0
        √ custom minting: should fail if qty = 0
        √ custom minting: should fail if not admin
        √ mintMultiple should fail if assets.length == 0
        √ mintMultiple should fail if catalystsQuantities == 0 (91ms)
        √ mintMultiple should fail if gemsQuantities == 0
        √ mintMultiple should fail if trying to add too many gems
        √ mintMultiple: should fail if gemsId = 6
        √ mintMultiple: should fail if catalystsId = 5
        √ mintMultiple: should not set catalyst if catalystId == 0 (1072ms)

  AssetUpgrader
    √ extractAndSetCatalyst for FT with rareCatalyst and powerGem, no ownership change (5342ms)
    √ extractAndSetCatalyst should fail for NFT (194ms)
    √ setting a rareCatalyst with powerGem and defenseGem (561ms)
    √ adding powerGem and defenseGem to a rareCatalyst with no gems (624ms)
    √ setting a rareCatalyst where ownerOf(assetId)!= msg.sender should fail (306ms)
    √ burns sand fees when feeRecipient = BURN_ADDRESS (502ms)

  CollectionCatalystMigrations
    √ migrating assetId with epic catalyst and no gems (6216ms)
    √ migrating assetId with rare catalyst and power gem (348ms)
    √ migrating assetId of quantity = 1 with legendary catalyst (356ms)
    √ migrating asset with collection id != 0 should fail (285ms)
    √ migrating assetId not from admin account should fail
    √ migrating assetId that does not exist in old registry should fail
    √ migrating assetId that has already been migrated should fail (343ms)
    √ batch migrating assetId not from admin account should fail
    √ batchMigrate two assets (451ms)
    √ setAssetAttributesRegistryMigrationContract first assignment (48ms)
    √ setAssetAttributesRegistryMigrationContract first assignment should fail for non admin
    √ setAssetAttributesRegistryMigrationContract second assignment (131ms)
    √ setAssetAttributesRegistryMigrationContract second assignment should fail for non migrationContract (81ms)

  GemsCatalystsRegistry
    √ getMaxGems for commonCatalyst should be 1 (3717ms)
    √ getMaxGems for non existing catalystId should fail
    √ burnCatalyst should burn 2 common catalysts from catalystOwner account (98ms)
    √ burnCatalyst should burn 2 common catalysts from superOperator account (104ms)
    √ Allow max value allowance for every gems and catalyst (155ms)
    √ Allow 0 allowance for every gems and catalyst (184ms)
    √ burnCatalyst should fail for unauthorized account
    √ burnCatalyst should fail for non existing catalystId
    √ burnCatalyst should fail for insufficient amount
    √ burnCatalyst should fail for account with no gems
    √ burnGem should burn 3 power gems from gemOwner account (104ms)
    √ burnGem should burn 3 power gems from superOperator account (108ms)
    √ burnGem should fail for unauthorized account
    √ burnGem should fail for non existing gemId
    √ burnGem should fail for insufficient amount
    √ burnGem should fail for account with no gems
    √ addGemsAndCatalysts should fail for existing gemId
    √ addGemsAndCatalysts should fail for existing catalystd
    √ addGemsAndCatalysts should add gemExample (56ms)
    √ addGemsAndCatalysts should add catalystExample (63ms)
    √ addGemsAndCatalysts should fail for gem id not in order
    √ addGemsAndCatalysts should fail for unauthorized user
    √ addGemsAndCatalysts should fail if too many G&C (173ms)
    √ addGemsAndCatalysts pass if max -1 G&C (157ms)
    √ burnDifferentGems for two different gem tokens (184ms)
    √ burnDifferentCatalysts for two different catalyst tokens (187ms)
    √ batchBurnGems for two different gem tokens and two different amounts (169ms)
    √ Change trusted forwarder  (67ms)

  MockLandWithMint.sol
    √ creation (375ms)
    √ cannot set polygon Land Tunnel to zero address (2751ms)
    √ supported interfaces (42ms)
    Mint and transfer full quad
      With approval
        √ transfers quads of all sizes (2358ms)
      Without approval
        √ reverts transfers of quads (1106ms)
      From self
        √ transfers of quads of all sizes from self (2329ms)
    Burn and transfer full quad
      √ should revert transfer quad from zero address
      √ should revert transfer quad to zero address
      With approval
        √ should not transfer a burned 1x1 quad (107ms)
        √ should not transfer burned quads (24768ms)
      From self
        √ should not transfer a burned 1x1 quad (104ms)
        √ should not transfer burned quads (24104ms)
    mint and check URIs
      √ mint and check URI 1 (75ms)
      √ mint and check URI 3 (60ms)
      √ mint and check URI 6 (91ms)
      √ mint and check URI 12 (224ms)
      √ mint and check URI 24 (742ms)
      √ reverts check URI for non existing token
    Mint and transfer a smaller quad
      √ transferring a 1X1 quad from a 3x3 (122ms)
      √ transferring a 1X1 quad from a 12x12 (269ms)
      √ transferring a 3X3 quad from a 6x6 (166ms)
      √ transferring a 6X6 quad from a 12x12 (308ms)
    Mint and transfer all its smaller quads
      √ transferring all 1X1 quad from a 3x3 (577ms)
      √ transferring all 1X1 quad from a 6x6 (2235ms)
      √ transferring all 1X1 quad from a 12x12 (8718ms)
    transfer batch
      √ transfers batch of quads of different sizes (1892ms)
      √ transfers batch of quads of different sizes from self (1868ms)
      √ reverts transfers batch of quads to address zero
      √ reverts transfers batch of quads from address zero
```

```
      √ reverts transfers batch of quads for invalid parameters
    Testing transferFrom
      √ Transfer 1x1 without approval (42ms)
      √ Transfer 1x1 with approval (135ms)
    testing batchTransferFrom
      √ Mint 12x12 and transfer all internals 1x1s from it (566ms)
    Meta transactions
      transferQuad without approval
        √ should not transfer quads of any size (5606ms)
      transferQuad with approval
        √ should transfer quads of any size (4035ms)
      transferQuad from self
        √ should transfer quads of any size (3629ms)
      Burn and transfer full quad
        √ should revert transfer of 1x1 quad after burn (144ms)
        √ should revert transfer of any size quad after burn (43886ms)
      batchTransferQuad
        √ should batch transfer 1x1 quads (254ms)
        √ should batch transfer quads of different sizes (996ms)
    Getters
      √ returns the width of the grid
      √ returns the height of the grid
      √ should fetch x and y values of given quad id (2425ms)
      √ cannot fetch x and y values of given non existing quad id
      √ should fetch owner of given quad id (2443ms)

  PolygonLand.sol
    Land <> PolygonLand: Transfer
      L1 to L2
        √ only owner can pause tunnels
        √ only owner can unpause tunnels
        √ set Max Limit on L1
        √ cannot set Max Limit on L1 if not owner
        √ set Max Allowed Quads (38ms)
        √ cannot Max Allowed Quads if not owner
        √ should not be able to transfer Land when paused (164ms)
        √ should be able to transfer 1x1 Land (130ms)
        √ should be able to transfer 3x3 Land (165ms)
        √ should be able to transfer 6x6 Land (256ms)
        √ should be able to transfer 12x12 Land (532ms)
        √ should be able to transfer 24x24 Land (1700ms)
        √ should should be able to transfer multiple lands (453ms)
        Through meta transaction
          √ should be able to transfer 1x1 Land (151ms)
          √ should be able to transfer 3x3 Land (186ms)
          √ should be able to transfer 6x6 Land (263ms)
          √ should be able to transfer 12x12 Land (502ms)
          √ should should be able to transfer multiple lands meta (285ms)
      L2 to L1
        √ only owner can pause tunnels
        √ only owner can unpause tunnels
DUMMY CHECKPOINT. moving on...
        √ should not be able to transfer Land when paused (291ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 1x1 Land (249ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 12x12 Land (927ms)
        √ should not be able to transfer 2, 12x12 Land at once (925ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 3x3 Land (308ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 6x6 Land (425ms)
DUMMY CHECKPOINT. moving on...
        √ should should be able to transfer multiple lands (456ms)
        √ should not be able to transfer if exceeds limit (179ms)
      Through meta Tx
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 1x1 Land (285ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 3x3 Land (378ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 6x6 Land (440ms)
DUMMY CHECKPOINT. moving on...
        √ should be able to transfer 12x12 Land (933ms)

  PolygonLandWeightedSANDRewardPool
    √ last time reward application should match the duration (2414ms)
    √ total supply is at first empty
    √ staking should update the reward balance, supply and staking token balance (137ms)
    √ withdraw should update the reward balance, supply and staking token (124ms)
    √ reward per token should be 0 if total supply is 0
    √ reward per token calculation (120ms)
    √ earned calculation (120ms)
    √ get reward should transfer the reward and emit an event (140ms)
    √ exiting should withdraw and transfer the reward (111ms)
    √ pool contains reward tokens (48ms)
    √ user can earn reward tokens if pool has been notified of reward (93ms)
    √ admin can notify to start a new reward process (without sending more reward tokens) (104ms)
    √ user cannot earn rewardTokens if they stake after the end time (79ms)
    √ user earns full reward amount if there is only one staker after 1 day(s) (93ms)
    √ user earns full reward amount if there is only one staker after 27 day(s) (106ms)
    √ User with 0 LAND earns correct reward amount (124ms)
    √ User with 0 LAND earns correct reward amount - smaller stake (123ms)
    √ User with 1 LAND(s) earns correct reward amount (168ms)
    √ User with 3 LAND(s) earns correct reward amount (182ms)
    √ User with 10 LAND(s) earns correct reward amount (307ms)
    √ User can withdraw some stakeTokens after several amounts have been staked (72ms)
    √ First user can claim their reward - no NFTs (134ms)
    √ First user can claim their reward - has NFTs (312ms)
    √ A user can claim their reward after multiple stakes (421ms)
    √ First user can exit the pool (109ms)
    √ A user can exit the pool after multiple stakes (156ms)
    √ A user with NFTs can exit the pool after multiple stakes (435ms)
    √ Change externals contracts
    √ user earnings for 0 NFT(s) match expected reward with 1 stake(s) (94ms)
    √ user earnings for 0 NFT(s) match expected reward with 2 stake(s) (94ms)
    √ user earnings for 0 NFT(s) match expected reward with 4 stake(s) (138ms)
    √ user earnings for 0 NFT(s) match expected reward with 10 stake(s) (219ms)
    √ user earnings for 1 NFT(s) match expected reward with 1 stake(s) (123ms)
    √ user earnings for 1 NFT(s) match expected reward with 10 stake(s) (423ms)
    √ user earnings for 2 NFT(s) match expected reward with 1 stake(s) (142ms)
    √ user earnings for 3 NFT(s) match expected reward with 1 stake(s) (171ms)
    √ user earnings for 3 NFT(s) match expected reward with 10 stake(s) (597ms)
    √ user earnings for 89 NFT(s) match expected reward with 1 stake(s) (1507ms)
    √ user earnings for 89 NFT(s) match expected reward with 10 stake(s) (1947ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 2 users, 10 stake each (431ms)
    √ Multiple Users' earnings for 0 NFTs match expected reward: 3 users, 1 stake each (139ms)
    √ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 1 stake each (180ms)
    √ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 10 stake each (213ms)
    √ Multiple Users' earnings for 3 NFTs match expected reward: 2 users, 1 stake each (265ms)
    √ Multiple Users' earnings for 100 NFTs match expected reward: 2 users, 1 stake each (3279ms)
    √ Staking with STAKE_AMOUNT plus an extra amount equivalent to 2 NFTs (126ms)
    √ Earlier staker gets more rewards with same NFT amount - small NFT number (141ms)
    √ Earlier staker gets more rewards with same NFT amount - large NFT number (3366ms)
    √ More lands give more rewards than earlier staker when NFT amounts are smaller (205ms)
    √ More lands do not give more rewards than earlier staker with large NFT amounts (3275ms)
    √ rewardToken in pool is more than amount notified (110ms)
    √ rewardToken in pool is zero (88ms)
    √ rewardToken in pool is less than amount notified (112ms)
    √ the call to notifyRewardAmount is made after users first call stake (98ms)
    √ user is earning rewards and pool is notified for a second time before end of current reward period (99ms)
    √ Multiplier & reward are correct (260ms)
    √ Only sender or reward distribution can compute sender's account

  PolygonSANDRewardPool
    √ last time reward application should match 30 days (1319ms)
    √ total supply is at first empty
    √ staking should update the reward balance, supply and staking token balance (93ms)
    √ withdraw should update the reward balance, supply and staking token (124ms)
    √ reward per token should be 0 if total supply is 0
    √ reward per token calculation (109ms)
    √ earned calculation (109ms)
    √ get reward should transfer the reward and emit an event (108ms)
    √ exiting should withdraw and transfer the reward (94ms)

  PolygonSand.sol Meta TX
    transfer
      √ without metatx (46ms)
      √ with metatx (95ms)
    approve and transferFrom
      √ without metatx (48ms)
      √ with metatx (171ms)
    burn
      √ without metatx (79ms)
      √ with metatx (140ms)
    trusted forwarder
      √ should fail to set the trusted forwarder if not owner
      √ should success to set the trusted forwarder if owner (62ms)
    approveAndCall
      √ without metatx (46ms)
      √ with metatx (109ms)
    paidCall
      √ without metatx (48ms)
      √ with metatx (111ms)

  PolygonSand.sol
    Bridging: L1 <> L2
      √ should update the child chain manager
      √ should fail if not owner when updating the child chain manager
      √ should fail when updating the child chain manager to address(0)
      √ should be able to transfer SAND: L1 to L2 (2653ms)
      √ should be able to transfer SAND: L2 to L1 (2761ms)
    Getters
      √ gets the correct name of the Sand Token
      √ gets the correct symbol of the Sand Token

  SandBaseToken.sol
    Deployment
      √ total supply should be 3,000,000,000 * 10^18 (420ms)
    Transfers
      √ users should be able to transfer some of the token they have (77ms)
      √ users should be able to transfer all token they have (77ms)
      √ users should not be able to transfer more token than they have
      √ users should not be able to transfer token they dont have
      √ users balance should not move if they transfer token to themselves
      √ total supply should not be affected by transfers (56ms)
    Allowance
      √ user should not be able to transfer more token than their allowance permit
      √ user should be able to transfer some of the amount permitted by their allowance
      √ user should be able to transfer all tokens that their allowance permit
      √ burn test

  PolygonSandClaim
    √ fetches the given amount of fake sand from user and transfers the same amount of new sand (1324ms)
    √ reverts if claim amount is more than balance
    √ returns the amount of sand which has been claimed (72ms)

  SandPolygonDepositor
    √ Locking funds in sand predicate mock contract (978ms)

  RaffleTheDoggies
    - should be able to mint with valid signature
    - should be able to mint 10_000 different tokens
    - should be able to mint 10_000 different tokens in 3 waves
    - should be able to mint 10_000 different tokens in 3 waves in 3 txs

  ERC20BasicApproveExtension
    √ ApproveAndCall calling buyLandWithSand (2743ms)
    √ ApproveAndCall should fail for input data too short (43ms)
    √ ApproveAndCall should fail for first parameter != sender
    √ ApproveAndCall should fail for zero data
    √ ApproveAndCall should fail for Approving the zeroAddress (45ms)
    √ ApproveAndCall should work for target = EOA with ether value = 1 (152ms)
    √ ApproveAndCall should work for target = EOA with ether value = 0 (156ms)
    √ ApproveAndCall for an empty contract as a target should revert
    √ ApproveAndCall calling logOnCall of a mock contract (119ms)
    √ ApproveAndCall with only one parameter should fail
    √ ApproveAndCall calling revertOnCall of a mock contract should fail (39ms)
    √ PaidCall calling buyLandWithSand (346ms)
    √ PaidCall should fail for input data too short (39ms)
    √ PaidCall should fail for first parameter != sender
    √ PaidCall should fail for zero data
    √ PaidCall should fail for Approving the zeroAddress (44ms)
    √ PaidCall should work for target = EOA with ether value = 1 (135ms)
    √ PaidCall should work for target = EOA with ether value = 0 (153ms)
    √ PaidCall for an empty contract as a target should revert
    √ PaidCall calling logOnCall of a mock contract (150ms)
    √ PaidCall calling revertOnCall of a mock contract should fail

  Gems & Catalysts permit
    √ user can use permit function to approve Gems via signature (61ms)
    √ user can use permit function to approve Catalysts via signature (62ms)
    √ updates a users allowances correctly
    √ should fail if deadline < block.timestamp
    √ should fail if recoveredAddress == address(0) || recoveredAddress != owner
    √ should fail if owner == address(0) || spender == address(0)

  Sand.sol
    Deployment
      √ total supply should be 3,000,000,000 * 10^18 (58ms)
    Transfers
      √ users should be able to transfer some of the token they have (59ms)
      √ users should be able to transfer all token they have (77ms)
      √ users should not be able to transfer more token than they have
      √ users should not be able to transfer token they dont have
      √ users balance should not move if they transfer token to themselves
      √ total supply should not be affected by transfers (56ms)
    Allowance
      √ user should not be able to transfer more token than their allowance permit
      √ user should be able to transfer some of the amount permitted by their allowance
      √ user should be able to transfer all tokens that their allowance permit
    Getters
      √ gets the correct name of the Sand Token
      √ gets the correct symbol of the Sand Token

  Batch.sol coverage
    √ atomicBatchWithETH (503ms)
    √ nonAtomicBatchWithETH
    √ atomicBatch
    √ nonAtomicBatch
    √ singleTargetAtomicBatchWithETH
    √ singleTargetNonAtomicBatchWithETH
    √ singleTargetAtomicBatch
    √ singleTargetNonAtomicBatch
    √ onERC1155Received
    √ onERC1155BatchReceived
    √ onERC721Received
    √ supportsInterface (62ms)


  1161 passing (9m)
  5 pending
```

Code Coverage

```
yarn coverage
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Game\	100	82.86	100	100	
GameBaseToken.sol	100	83.33	100	100	
GameMinter.sol	100	75	100	100	
GameV1.sol	100	100	100	100	
Sand\	100	100	100	100	
SandBaseToken.sol	100	100	100	100	
Utils\	100	50	100	100	
Batch.sol	100	50	100	100	
asset\	86.73	66.3	83.21	86.71	
AssetAttributesRegistry.sol	97.33	83.33	100	97.47	170,171
AssetMinter.sol	96	80.77	91.67	96.1	216,218,334
AssetSignedAuctionAuth.sol	75.68	59.09	64.29	75.34	… 304,398,403
AssetUpgrader.sol	93.02	65.38	84.62	93.02	219,221,229
AssetV2.sol	93.75	50	100	93.75	45
ERC1155ERC721.sol	83.28	62.73	80	83.12	… 795,907,908
asset\libraries\	100	75	100	100	
AssetHelper.sol	100	70	100	100	
ERC1155ERC721Helper.sol	100	100	100	100	
bundleSandSale\	100	67.39	100	100	
PolygonBundleSandSale.sol	100	67.39	100	100	
catalyst\	99.06	95.45	97.06	99.06	
Catalyst.sol	100	100	100	100	
CollectionCatalystMigrations.sol	100	91.67	100	100	
DefaultAttributes.sol	100	100	100	100	
Gem.sol	100	100	100	100	
GemsCatalystsRegistry.sol	98.31	96.15	95.65	98.31	252
catalyst\interfaces\	100	100	100	100	
ICollectionCatalystMigrations.sol	100	100	100	100	
IGemsCatalystsRegistry.sol	100	100	100	100	
IOldCatalystRegistry.sol	100	100	100	100	
claims\AssetGiveaway\	96.67	93.75	90.91	96.67	
AssetGiveaway.sol	91.67	87.5	80	91.67	72
ClaimERC1155.sol	100	100	100	100	
claims\MultiGiveaway\	97.96	96.43	94.44	97.96	
ClaimERC1155ERC721ERC20.sol	100	92.86	100	100	
MultiGiveaway.sol	96.3	100	91.67	96.3	131
claims\signedGiveaway\	96.67	92.86	91.67	96.67	
SignedERC20Giveaway.sol	96.67	92.86	91.67	96.67	133
common\BaseWithStorage\	85.03	65.96	85.25	85.14	
ERC2771Handler.sol	62.5	50	80	66.67	36,37,39
ERC721BaseToken.sol	**83.96**	**67.65**	**86.21**	**84.55**	**… 382,384,399**
ImmutableERC721.sol	96	66.67	90	96	80

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
MetaTransactionReceiver.sol	40	0	33.33	40	14,15,27
WithAdmin.sol	100	75	100	100	
WithMinter.sol	100	100	100	100	
WithPermit.sol	100	100	100	100	
WithSuperOperators.sol	100	50	100	100	
WithUpgrader.sol	75	0	66.67	60	15,16
common\BaseWithStorage\ERC20\	93.85	71.05	90.91	93.85	
ERC20BaseToken.sol	93.55	71.05	89.47	93.55	120,148,149,150
ERC20Token.sol	100	100	100	100	
common\BaseWithStorage\ERC20\extensions\	100	50	100	100	
ERC20BasicApproveExtension.sol	100	50	100	100	
ERC20Internal.sol	100	100	100	100	
common\BaseWithStorage\ERC677\extensions\	100	100	100	100	
ERC677Extension.sol	100	100	100	100	
common\Base\	100	100	100	100	
TheSandbox712.sol	100	100	100	100	
common\Libraries\	82.22	52.63	93.33	81.32	
BytesUtil.sol	75	50	100	80	13
ObjectLib32.sol	95.24	80	100	95	52
PriceUtil.sol	62.5	50	100	57.14	18,21,25
SafeMathWithRequire.sol	100	50	100	100	
SigUtil.sol	45	28.57	66.67	45.45	… 43,46,47,49
Verify.sol	100	100	100	100	
common\interfaces\	100	100	100	100	
ERC1271.sol	100	100	100	100	
ERC1271Constants.sol	100	100	100	100	
ERC1654.sol	100	100	100	100	
ERC1654Constants.sol	100	100	100	100	
IAssetAttributesRegistry.sol	100	100	100	100	
IAssetMinter.sol	100	100	100	100	
IAssetToken.sol	100	100	100	100	
IAssetUpgrader.sol	100	100	100	100	
IAttributes.sol	100	100	100	100	
IERC1155.sol	100	100	100	100	
IERC1155TokenReceiver.sol	100	100	100	100	
IERC165.sol	100	100	100	100	
IERC20.sol	100	100	100	100	
IERC20Extended.sol	100	100	100	100	
IERC677.sol	100	100	100	100	
IERC677Receiver.sol	100	100	100	100	
IERC721.sol	100	100	100	100	
IERC721Events.sol	100	100	100	100	
IERC721Extended.sol	100	100	100	100	
IERC721MandatoryTokenReceiver.sol	100	100	100	100	
IERC721TokenReceiver.sol	**100**	**100**	**100**	**100**	
IGameMinter.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IGameToken.sol	100	100	100	100	
ILandToken.sol	100	100	100	100	
IPolygonLand.sol	100	100	100	100	
Medianizer.sol	100	100	100	100	
defi\	94.12	81.82	92.31	93.33	
SandRewardPool.sol	93.68	81.82	91.67	92.86	… 172,235,385
StakeTokenWrapper.sol	100	100	100	100	
defi\contributionCalculation\	95.24	80	90	95.24	
LandContributionCalculator.sol	92.31	83.33	83.33	92.31	36
LandOwnerContributionCalculator.sol	100	75	100	100	
defi\interfaces\	100	100	100	100	
IContributionCalculator.sol	100	100	100	100	
IRewardCalculator.sol	100	100	100	100	
defi\rewardCalculation\	80.68	83.33	84.62	80.68	
PeriodicRewardCalculator.sol	100	100	100	100	
TwoPeriodsRewardCalculator.sol	72.58	76.67	76.47	72.58	… 176,177,178
faucet\	95.45	75	100	95.45	
Faucet.sol	95.45	75	100	95.45	89
permit\	100	100	100	100	
Permit.sol	100	100	100	100	
polygon\LiquidityMining\	91.38	63.64	93.18	91.67	
IRewardDistributionRecipient.sol	100	75	100	100	
PolygonLandWeightedSANDRewardPool.sol	95.59	70.83	95.83	95.65	226,230,232
PolygonSANDRewardPool.sol	84.44	50	88.24	84.78	… 145,149,151
polygon\child\	100	100	100	100	
ChildGameTokenV1.sol	100	100	100	100	
polygon\child\asset\	95.24	62.5	100	95.24	
PolygonAssetV2.sol	95.24	62.5	100	95.24	46
polygon\child\land\	87.83	68.89	93.33	88.93	
PolygonLandBaseToken.sol	87.01	68.45	100	87.97	… 565,569,570
PolygonLandTunnel.sol	90.7	75	82.35	92.68	97,131,140
PolygonLandV1.sol	100	75	100	100	
polygon\child\sand\	95.24	70	90.91	95.24	
PolygonSand.sol	91.67	66.67	85.71	91.67	55
PolygonSandClaim.sol	100	75	100	100	
polygon\child\sand\interfaces\	100	100	100	100	
IPolygonSand.sol	100	100	100	100	
polygon\root\	100	100	100	100	
IRootChainManager.sol	100	100	100	100	
SandPolygonDepositor.sol	100	100	100	100	
polygon\root\land\	86.96	50	76.92	86.96	
LandTunnel.sol	85	50	72.73	85	34,70,97
MockLandTunnel.sol	100	100	100	100	
raffle\	0	0	0	0	
Raffle.sol	0	0	0	0	… 217,221,225
All files	**88.4**	**69.18**	**87.92**	**88.53**	

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
98f3b08790b5e5f5f8bbd00bf699b29ea30bf0017c73250738ce24b8127a6226 ./ERC2771Handler.sol

70e28124291b77fbcff215a4fea8eb0a0726ed444ae2234284aa71227f55d171 ./ERC721BaseToken.sol

9073b3da579f093123bf2419fef081f9aad801b46b6170d52dc49e0752fce800 ./IERC721MandatoryTokenReceiver.sol

aae5dcb417dec176726e5ade31275f71f5105ba336b8497ae7b9e7b88072c11f ./ILandToken.sol

cfafb021c4ac9c2e024d6164c946e42f25a30c4ab96972b865c53c424ddec6b2 ./IPolygonLand.sol

78fe5432325d8e4010ad4c0b2c91a6764f1f4a776473cdc825b05d25b82fa7e4 ./LandTunnel.sol

6c4d8a9017f4708c0dbcca0c794c131e57df0a7a61783ef54427a2766f207b3d ./PolygonLandBaseToken.sol

9b424f87d48a620c4b06b08ace25ce3a37f992095c8d61f73674fd5ca151a3d6 ./PolygonLandTunnel.sol

df7bb1183a51a729822e8309fc123194e9bb12b1d600d6f81b52e5788f619262 ./PolygonLandV1.sol
```

Tests

```
4e2585a6ddb2147ba235c09d16feee3b1e16264ccbfbf04b7ebf9a17b248c841 ./test/chai-setup.ts

202266a6189fcff67bcc7ce9e59a463de535fa4d4a6798e6a1b16b75ca2c6ad8 ./test/chaiBigNumberCloseTo.ts

ce685453a3762463b449bfb92619a55245b0f683e2379e9e4796f23ebc3bc8f3 ./test/evmSnapshotRevert.test.ts

c1b5c8c0197083d7216bab7d946557171356e5546f90b26209eec31240c888be ./test/forwardRequestData712.ts

9e32f60dfdfc031ca589cf789f48458a023533a1d87d38894471f134123e513c ./test/sendMetaTx.ts

3ac8b5c4f8df8821ef6d9a9264c132bc50a33eba5a64935957520ea9b9b0588a ./test/utils.ts

c4c9b763c555ac9d378b788a24f93d60e08ebaf87f99bac176bc56eb35c12121 ./test/utils/batch.test.ts

2b61b76c0976d7920ec16807d6dd2d2d8ad49b44c731caeca1192e112f93d8bb ./test/sand/erc20BasicApproveExtension.test.ts

52ead9788e626791cf23b33d5753d2e2c17858b1a471d66d2f3dd9a15d407325 ./test/sand/ERC20Permit.test.ts

7beff9569c7bd158ae5fad2a58fb0d2608764a2ff23d427ff36bd5c10b254a98 ./test/sand/fixtures.ts

a1de6ff93ae1a906d4ab84863a371deda13fc277c275f6bd28b204aa52d11833 ./test/sand/Sand.test.ts

d519a8acc9cb619ed1ee4bd47fc7845a37453010bd73f43b0eeab135e857aa5e ./test/raffle/fixtures.ts

129366cff63ba13e162d8f09a03c617f1db23cc6710f173f73672d632b15acee ./test/raffle/raffle.test.ts

594831422c7334078301c453e877e2bc8f10ccd6e7c1cdc6776b364c1cd5ee94 ./test/polygon/sandPolyonDepositor/fixtures.ts

bac73da75a5899d168d1ce67cdde705127d2120bc12efc549542c8b8b8b7a640 ./test/polygon/sandPolyonDepositor/sandPolygonDepositor.test.ts

042216994b54f28c3110930d1a28ca5a3e88d45fdbe997d8f00f258be1f63213 ./test/polygon/sandClaim/fixtures.ts

ccf30de50eb5a833eb30840eb90f28a0486fb900dc6243cf43ede72751d49de8 ./test/polygon/sandClaim/polygonSandClaim.test.ts

6956ac6aeba350418c4fa37722ff5865480e6ef1ffaa23910e1f5877764240ff ./test/polygon/sandBase/sandbase.test.ts

fd36f25cf913392c8e5e83c06c4c90ed7face7cab3e448fb180ffd1f136f9bb6 ./test/polygon/sand/fixtures.ts

95541f43f6f5f5bc0f7132f71d97ca01000a2e5de20c04fcc00f301f645f1e2e ./test/polygon/sand/sand-metatx.test.ts

6f601c7ee566878b3d863cb19d5e07305295e1fa1d6aa97dbf1df926135be067 ./test/polygon/sand/sand.test.ts

c16f09e9c8e954db6e3cab94525262ab3a2ec6e2c5b5b62a19d33ad7ae9889b0 ./test/polygon/liquidityMining/fixtures.ts

ca83cdcb353f7df968f7f426cd1cbacd8fe72f5e7b77658ba8f3aaf4a2ecf765 ./test/polygon/liquidityMining/polygonLandWeightedSANDRewardPool.test.ts

7cae5c9423b098dbcc4f7cda8aa82b6332f7765fdd491dfdfa68b8fb4d2739b9 ./test/polygon/liquidityMining/polygonSANDRewardPool.test.ts

0ad6d0adedd43e6e0189fe5b0456436fbbb48d559e28d2cf077bb33f10c4f207 ./test/polygon/land/fixtures.ts

25e9d12d7a4ee67fa961957c9eca97f02ebf34740095151a1d082f9d7395fd18 ./test/polygon/land/land.test.ts

2a0d603330a3592455cef6a07e1ee2986ccbfba8973050ffcc114506449394d7 ./test/polygon/land/landTunnels.test.ts

7fcfb0818fb12bc1594fee46c1d80013ba9f3b676491338d01a7d16755ce7c3e ./test/polygon/land/testERC721.test.js

4a91af4128d0bf0753e368d4e080650017e3c291e76d310cc79172effd650ae6 ./test/polygon/catalyst/utils.ts

e656182bc11b5ab3b9c28cc63f7550a6b81e38918a2995ad6a6bf5fc63301cdf ./test/polygon/catalyst/gemsCatalystsRegistry/catalyst.test.js

526dd82dbd0d9703d2c230ac011d09d3ac3bc64e60146d7be22de04166e0899c ./test/polygon/catalyst/gemsCatalystsRegistry/gems.test.js

bab1f8fac31b984cdec01654ba26bc9edff1de9d56cff4fcafe1ba2af03d4a47 ./test/polygon/catalyst/gemsCatalystsRegistry/gemsCatalystsRegistry.test.ts

715c6db78c401e2e87ccb91b89afc764abd3af777441cb44c23b80b7bd58d202
./test/polygon/catalyst/collectionCatalystMigrations/collectionCatalystMigrations.test.ts

e10cb32b373e7d01f4b00bd9b8854191e2ba619adb727aae293ce82de5189f09 ./test/polygon/catalyst/collectionCatalystMigrations/fixtures.ts

a686099adb5d4801f4f08d716cf64c51b4c0fa08d76f8e241def386f92e0b36a ./test/polygon/catalyst/assetUpgrader/assetUpgrader.test.ts

c5d562b23d56d7b0894d273fe35b2b60920669378d511e216e66c148c8b39c52 ./test/polygon/catalyst/assetMinter/assetMinter.test.ts

b7534e1690bdbfbb45926bc7b13f38e62844e9202376e802870c2ad1966f32c0 ./test/polygon/catalyst/assetMinter/fixtures.ts

45f45ebebe2a1417f06bf301783219b5df34cfaa84eff59731015eb5ace9b412
./test/polygon/catalyst/assetAttributesRegistry/assetAttributesRegistry.test.ts

8756addece7034f0351744f294a03136400d82b1dbf5191604122ae84123e46d ./test/polygon/catalyst/assetAttributesRegistry/fixtures.ts

b9f79ac7d46caa62afa0c71ff59cab36feba1ed05cb0b0532a94526ed322329e ./test/polygon/catalyst/assetAttributesRegistry/getAttributes.test.ts

268ca8c41c8bc0a82a4b5cbcb17d29bd60c47c19a4c6230a70763aa2bf48b909 ./test/polygon/bundleSandSale/bundleSandSale.test.ts
```

```
b806141f82e8d0ba332beadc5018e2f7265463ae2e6d9c5d4aff9a0d0b5fcf52 ./test/polygon/bundleSandSale/fixtures.ts
949561f53e85e19ca03fdcf788dd257f19f87810409d01cee061e502e7495c5f ./test/polygon/asset/asset.test.ts
fb4bf684456cba08a7e17b94fc36e4c2bbcb209be514c0b699b14ae2afc76ef8 ./test/polygon/asset/assetSignedAuctionAuth.test.ts
c1b2179c883d675d50c3e26cbfc84328ce1668b797f6b8f608742151f6c1ea61 ./test/polygon/asset/fixtures.ts
2ef7c4c68fec114c2718812fa7a3f5816ee7752072b4b61fca29895a227f8e77 ./test/permit/data712.ts
cb019cdadcb0720c95436db9dc00799227d1a201e7c26980b7117cc79abc1b7d ./test/permit/fixtures.ts
e4a05f8b305350caf07ec7a0fc83423ae78c58b3bb832842e1929907f8bbb140 ./test/permit/permit.test.ts
d8337ff0c027f771b549306859151a60ffc86978aefe4bfa283a4c9fd3a1a26b ./test/multiGiveaway/balanceHelpers.ts
110bab7780b9f5f64d859f77fb15b82ab549798907f67f7fe75d8b57ce10cf8f ./test/multiGiveaway/fixtures.ts
ea43bf3dcf7acd814bcc92f89cee2eec8eb9110fd954e7c6df2ee1d737ff55b5 ./test/multiGiveaway/gas.test.ts
40119487b7614d062cce7fab896f84c2b989c0f2939f855b48a3e21b9eb38811 ./test/multiGiveaway/merkleTree.test.ts
690490ae892f516a5c2e227bdfd8e2dfcdf8f92cca52785b8eb5ae749daa3570 ./test/multiGiveaway/multiGiveaway.test.ts
03d7855ef6e22c20254d7bf63f190365bb17873646734bf8c12038dc207e0e98 ./test/liquidityMining/contributionEquation.test.js
4c1bd4e849b1d1527e8760bdd197727d53c0782f63c1ab8b4ef3106c42450f67 ./test/liquidityMining/mockSANDRewardPool.test.js
11d7ac4fb2c753e8fbadafbaa312597129dc2a2478f3ec62f18f1e9afa23733c ./test/liquidityMining/SANDRewardPool.test.js
5ae7b1e3b742b624b07e2c84a9fd7dba1a1815801b1cc1a98350ff7754c3603e ./test/liquidityMining/_testHelper.js
4ef0f64fe9b9c4f95f81ca4aefc80e68ed8251abe92f4a86b72c122feaa3385b ./test/land-sale/AuthValidator.test.ts
0f6f777865707c6088160cfc875d59fc649324ad21154adf63f255647f993c57 ./test/land-sale/EstateSaleWithAuth.test.ts
ebe923755f90f8ca572637f44f119bed0614c41abc91fde6cd4a8fb17f3c46f1 ./test/land-sale/fixtures.ts
5a10830aae480e06b6e0643c55d2b8a47cdc1c4e1a1c903c7575cbd401e65dc5 ./test/Game/assets.ts
c1b5c8c0197083d7216bab7d946557171356e5546f90b26209eec31240c888be ./test/Game/data712.ts
1b49885663e44edf917c834415cfc94eb858d07de746fb5f68604c940d949c7f ./test/Game/fixtures.ts
3fc6933b88240bb58690742350745be58e7d6e7315401ccd27bacceb8c9148e7 ./test/Game/GameToken.test.ts
0a1f020ca8a1426dd717fbf7de6699f756f1e53d7dd35cc581083d37d9b74245 ./test/Game/testERC721.test.js
5330049eb83769808a63d13d52f80c19e0f3af96bf6b334ffb116234628c7146 ./test/Game/gameMinter/gameMinter.test.ts
6d00a2d48163c5a504702f421b872bd258f4294174fd9092ac180b9140ad3d0b ./test/faucet/faucet.test.ts
75515bd70629a9f2533fdd11be2ae7ec5216734419f84dbbf00b6cc1f34fe122 ./test/faucet/fixtures.ts
99819c1d28bdb6c2bc782825126d365105a12f62846730fe81ea80cc9f1a5580 ./test/erc721/index.js
b23a4d22756f3e776c87be1a9fb66267a9e90bc0e7189345c8408a9f791e7764 ./test/erc677/ERC677.test.ts
31756ce78a0b647e4ff5a0f494db0685020954113d63593cf61149aa5febd4f7 ./test/erc677/fixtures.ts
306e15dbbf40f2dcbd3442d9fb1825b92cfb5a4153c003533112d514ee934a8d ./test/erc20/index.js
bdff7e20506d8e2c47e387f1e13aa75fb32bf0bd194ca43818d8a8e82cdcd081 ./test/erc1155/index.js
c38b3efc3aeab6823ba606c2d4e6631baf11db822d617c072d960e40986d81f4 ./test/defi/sandRewardPool/originalSet.test.ts
f4fb9c8cc47ec789b9731ed8e8b870d53b8da01870cb7aa8bdf86133632ea2e2 ./test/defi/sandRewardPool/sandRewardPool.e2e.test.ts
9d6e3c49ceea136e876dd2d26cc8ff451d0201cade1d758e26d2e35ac7430e53 ./test/defi/sandRewardPool/sandRewardPool.test.ts
082b4da1ad32bfeffffeee59d97849a0603dacb38c1dd90727ab2877c91e630c ./test/defi/sandRewardPool/sandRewardPoolAntiCompund.test.ts
c2e31c1e43465663db1f22325878f30a02da46d382118321d08ec9c24474a17e ./test/defi/sandRewardPool/utils.ts
ab61d477415ac2c7ccb4950bc62b885d01acece29213aae4e2f834b84e62b4ec ./test/defi/sandRewardPool/fixtures/contributionCalculator.fixture.ts
aa7353b2c49a3dcffad0a9ebdf4e4a0f61aaba07a46a9ddc0120d84427108edf ./test/defi/sandRewardPool/fixtures/originalFixtures.ts
f71066f11fd8a404d783d5e7513facd61df4b01a562ff7f6eb2335d37311c320 ./test/defi/sandRewardPool/fixtures/rewardCalculator.fixture.ts
87717154cdbe0797a6f3a88c154365cbcf845b2716b12249914cbc5cfff0212c ./test/defi/sandRewardPool/fixtures/sandRewardPool.fixture.ts
dce71f8644f7a3b55a3764b895b5627b0ece6c6a494abac9e734aa68533d563d ./test/defi/sandRewardPool/calculators/landContributionCalculator.test.ts
19dc982906fd0afb2f5060eac9756aaf9494c9c46f88a17898038b18b908655e
./test/defi/sandRewardPool/calculators/landOwnerContributionCalculator.test.ts
2b209c3326eb06987dc7e22d5453547ac11bd3138a5609d4d5dbe31a15237625 ./test/defi/sandRewardPool/calculators/periodicRewardCalculator.test.ts
8e3c3f3a043f3bf957fdaa43a04a85eda8ced5b42f0bc04f870c2af7fe3a1ef1 ./test/defi/sandRewardPool/calculators/twoPeriodsRewardCalculator.test.ts
e1097fde2100ae26f0abd99b9794b9f4726831f112846c4e8101c488882d143c ./test/common/contributionEquation.ts
f5461b339b1849ced82083e51c9d71fd262a0de851159e5b722a4fd467a23951 ./test/common/libraries/safeMathWithRequire.ts
9e558932f7edda131934e45143585708c219c09221c0e7466209454b7a47764e ./test/common/fixtures/asset.ts
c3145d379e9c168a721a4b8c7cdd94ab227a9e7523b734e95d6452a5c41fae8b ./test/common/fixtures/assetAttributesRegistry.ts
beb6ef837233134200d6ac663cbe1d89c7317529f0b7e41c2676c1883d705d47 ./test/common/fixtures/assetUpgrader.ts
ce7090737921ca5fa6db7355539e8c3165dfefde7b1c88a17daed0da5f1ebbca ./test/common/fixtures/erc20BasicApproveExtension.ts
003546a9d771e98cb5253e9776ae5d807c254ba583e7866ee7ce2714158515c0 ./test/common/fixtures/gemAndCatalysts.ts
ee022b3c676541fc4662d00f93c7b29d842c5cd73035301aa20a1ecfa676ce7d ./test/claim/signedGiveaway/fixtures.ts
ca8447e666f25ec322cb764b4a36e580066bc242bf8485dfe2754031604d6f9e ./test/claim/signedGiveaway/signature.ts
c83ef66b3c535dbb007057ab4fc3bb49fcdd9224af66e89d7f8597e4ca44c1e0 ./test/claim/signedGiveaway/signedGiveaway.ts
0e631c666c0ed83c69931627790dcfb40f24f98b0b21ce183377894ebb88f818 ./test/assetGiveaway/assetGiveaway.test.ts
4eda3647ca32d173cf02e1f869784ec5c9c11d46cb143aabf197c9d7c8cb9186 ./test/assetGiveaway/fixtures.ts
01ee014a7bb074dc88f10485612bc362faaf994019685f2f51de63852574452a ./test/assetGiveaway/gas.test.ts
cbcb3b04e57d4056958eb618bc4d1c67ba7d6fe91bdb59561f210215d34e7f1c ./test/assetGiveaway/merkleTree.test.ts
c345a3919317d6a7a9f4e72434191e50e8d45fb5e8f3d8658b6eda161ca229d3 ./test/asset/asset.test.ts
```

```
8f2befda78ff078531afaab2cd9af3370c6daf6941559603de57e7e83f6d9ea5 ./test/asset/assetERC1155.test.js
cdd54628ab27ef887e9e3d9211e751a2b30df4b8d93a30c217ff544a98bb6907 ./test/asset/assetERC721.test.js
```

Changelog

- 2022-04-05 - Initial report
- 2022-05-19 - Final report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.