Quantstamp Security Assessment Certificate

November 10th 2022 — Quantstamp Verified

Sandbox Land

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

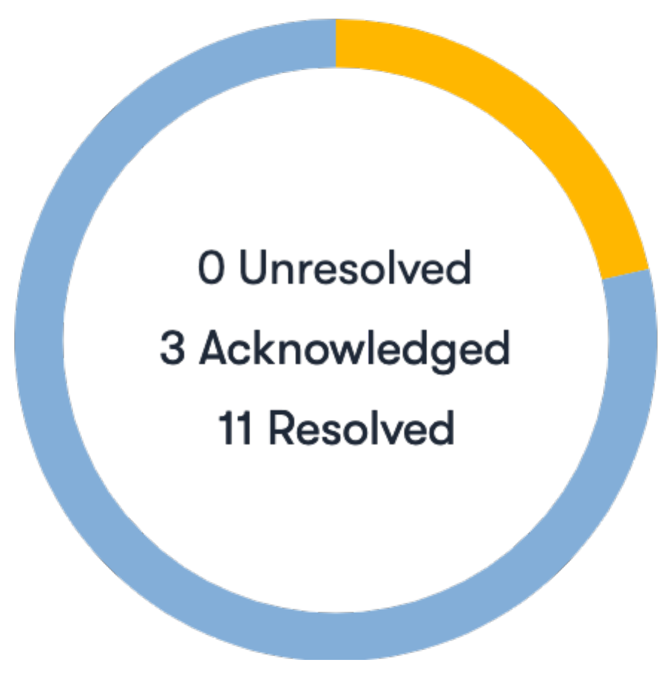
QUANTSTAMP VERIFIED · SECURITY CERTIFICATE ·

Executive Summary

Type	NFT
Auditors	Faycal Lalidji, Senior Security Engineer Marius Guggenmos, Senior Research Engineer Nikita Belenkov, Security Auditor
Timeline	2022-10-17 through 2022-10-24
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Readme.md Documentation
Documentation Quality	Medium
Test Quality	Low
Source Code	

Repository	Commit
https://github.com/thesandboxgame/sandbox-smart-contracts/tree/e4c40676218 1f243a6168279ec80548cfcf 43814/src/solc_0.6	e4c4067 initial audit

Total Issues	**14** (11 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	**4** (4 Resolved)
Informational Risk Issues	**8** (6 Resolved)
Undetermined Risk Issues	**1** (0 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
We have found multiple issues ranging from medium to undetermined severity that need to be fixed before deployment. We highly recommend implementing sufficient testing to possibly verify all the project specifications.
Fix review:
All highlighted issues have been either fixed or acknowledged.

ID	Description	Severity	Status
QSP-1	Reentrancy Is Possible During Token Transfer	^ Medium	Fixed
QSP-2	Missing Input Validation	⌄ Low	Fixed
QSP-3	Signature Malleability	⌄ Low	Fixed
QSP-4	Grace Period for Signing Wallet Represent a Risk	⌄ Low	Fixed
QSP-5	`ReferralValidator.handleReferralWithETH()` Does Not Verify `msg.value`	⌄ Low	Fixed
QSP-6	Single Step Admin Changes	○ Informational	Acknowledged
QSP-7	`transfer` Might Run Out of Gas	○ Informational	Fixed
QSP-8	Not Handling All Tokens Correctly	○ Informational	Fixed
QSP-9	Clone-and-Own	○ Informational	Fixed
QSP-10	Inherited Contract Is Not Fully Used	○ Informational	Fixed
QSP-11	Outdated Solidity Version	○ Informational	Acknowledged
QSP-12	Application Monitoring Can Be Improved by Emitting More Events	○ Informational	Fixed
QSP-13	Missing License	○ Informational	Fixed
QSP-14	Incorrect Use of Buyer Address in `EstateSaleWithAuth`	? Undetermined	Acknowledged

Quantstamp Audit Breakdown

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

DISCLAIMER:
If the final commit hash provided by the client contains features that are not within the scope of the audit or an associated fix review, those features are excluded from consideration in this report.

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 v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Reentrancy Is Possible During Token Transfer

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `ReferralValidator.sol`

Description: Currently, `EstateSaleWithAuth.buyLandWithSand()` does not use the `nonReentrant` modifier. This allows a potential reentrancy since an external call (`ERC20.transferFrom()`) is made when executing `EstateSaleWithAuth._handleFeeAndReferral()`.
Even trusted ERC20 implementation can allow reentrancy by executing random code in the receiver address (in the case of a contract).

Recommendation: Use a reentrancy guard in all required functions.

Update: Fixed in commit "47b3478d3cfaf2ed1d276fcd976fe975f68b60c2".

QSP-2 Missing Input Validation

Severity: *Low Risk*

Status: Fixed

File(s) affected: `MetaTransactionreceiver.sol`, `AuthValidator.sol`, `ReferralValidator.sol`, `EstateSaleWithAuth.sol`

Related Issue(s): SWC-123

Description: It is important to validate inputs to avoid human error, even if they only come from trusted addresses.

- `MetaTransactionreceiver._setMetaTransactionProcessor()` does not validate the processor address.
- `AuthValidator.constructor()` does not validate the admin address.
- `ReferralValidator.constructor()` does not validate any of the input parameters.
- `ReferralValidator.updateSigningWallet()` does not validate the input address.
- `ReferralValidator.handleReferralWithETH()` must validate `commissionRate` to be lower than `10000`.
- `EstateSaleWithAuth.constructor()` does not validate any of the input parameters.

Recommendation: We recommend adding the relevant checks.

Update: Fixed in commit "f1ec408c392a2e24b63722c243a00f5ad091390b".

QSP-3 Signature Malleability

Severity: *Low Risk*

Status: Fixed

File(s) affected: `SigUtil.sol`

Related Issue(s): SWC-117

Description: The given implementation of signature verification using `ecrecover` in `SigUtil` library is prone to signature malleability.

Recommendation: Consider using a secure wrapper like OpenZeppelin's ECDSA utility library, which performs additional security checks on the signature parameters.

Update: SigUtil.sol has not been fixed but the relevant usages have been replaced by OpenZeppelin's `ECDSA.recover()` in commit "57fd1acae5b52bcca7ab24b6780dd0855c17ba76".

QSP-4 Grace Period for Signing Wallet Represent a Risk

Severity: *Low Risk*

Status: Fixed

File(s) affected: `ReferralValidator.sol`

Description: In `ReferralValidator.updateSigningWallet()`, the previous wallet is given a grace period. However, this behavior will not allow disabling a compromised signing wallet address.

Recommendation: We recommend clearly documenting this feature and adding a function that disables any previous signing address in the case of a hack.

Update: 'ReferralValidator.disablePreviousSigningWallet()' function has been added to allow the owner to disable previous wallets in commit "4fb8436d3189d61ad69042a35958732b2cd8bdf5".

QSP-5 `ReferralValidator.handleReferralWithETH()` Does Not Verify `msg.value`

Severity: Low Risk

Status: Fixed

File(s) affected: `ReferralValidator.sol`

Description: `ReferralValidator.handleReferralWithETH()` does not verify if `msg.value` is sufficient, meaning that a user can use an amount that is higher than the actual ETH deposit to manipulate the contract transfer output.

Recommendation: Depending on how `ReferralValidator.handleReferralWithETH()` will be used, a check should be added to verify if `msg.value` is equal to or greater than `amount`. Verification must always be executed at the same function level to not introduce any future vulnerability.

Update: Fixed in commit "9c7d06cec2bc5ba5a0167f72ce4efb54c435d523".

QSP-6 Single Step Admin Changes

Severity: Informational

Status: Acknowledged

File(s) affected: `Admin.sol`

Description: When changing the admin via the `changeAdmin()` function, a mistake in the supplied `newAdmin` argument might lead to losing the admin functionality of the contract. A safer way to do this is in a two-step process:

1. Propose the new admin; the old one is still the admin.
2. The new admin claims the role of admin. The old admin is stripped of the admin rights and replaced by the new admin.

This way, only addresses that are under the original admin's control are valid.

Recommendation: Consider implementing a two-step process for changing the admin role.

Update: The project team acknowledged this issue.

QSP-7 `transfer` Might Run Out of Gas

Severity: Informational

Status: Fixed

File(s) affected: `ReferralValidator.sol`

Description: Use of the `address.transfer()` function is discouraged, since this function forwards only 2300 gas, which might break some non-trivial fallback functions if any future change increases the gas cost of opcodes operations.

Recommendation: Replace calls to `transfer()` with raw `call.value(amount)("")` calls and make sure to check the return value of these calls for success and to respect the Checks-Effects-Interactions Pattern.

Update: Fixed in commit "843435e34dd278d63cdffedae3795ea7020185f4".

QSP-8 Not Handling All Tokens Correctly

Severity: Informational

Status: Fixed

File(s) affected: `ReferralValidator.sol`

Description: The function `handleReferralWithERC20()` uses `ERC20.transferFrom()` calls wrapped in a require statement. However, some tokens do not implement the ERC20 interface correctly and do not return a value. Transfers will always fail if such a token is used for referral payments.

Recommendation: Consider using a library such as `OpenZeppelin's SafeERC20`, which handles such cases properly.

Update: Fixed in commit "798c48d04c9da90bf2735cd2971bbe7e9e9a009d".

QSP-9 Clone-and-Own

Severity: Informational

Status: Fixed

File(s) affected: `ERC20.sol`, `ERC1155.sol`

Description: The clone-and-own approach involves copying and adjusting open-source code at one's own discretion. From the development perspective, it is initially beneficial as it reduces the amount of effort. However, from the security perspective, it involves some risks as the code may not follow the best practices, may contain a security vulnerability or may include intentionally or unintentionally modified upstream libraries.

Recommendation: Rather than the clone-and-own approach, good industry practice is using a package manager (e.g. npm) to handle library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as using libraries. If the file is cloned anyway, a comment including the repository, the commit hash of the version cloned, and the summary of modifications (if any) should be added. This helps to improve the traceability of the file.
In the current case, `Interfaces/ERC20.sol` and `Interfaces/ERC1155.sol` could be replaced with the OpenZeppelin implementations.

Update: Fixed in commit "28ae473ce5937a91a9e34078ab7fb8132cf7b40f".

QSP-10 Inherited Contract Is Not Fully Used

Severity: Informational

Status: Fixed

File(s) affected: `ReferralValidator.sol`, `MetaTransactionReceiver.sol`

Description: `MetaTransactionReceiver` and `ReferralValidator` contracts inherit from the `Admin` contract, which controls the system's current Admin. But they don't use the modifier that has been inherited and reimplement the same logic in the following functions:

1. `MetaTransactionReceiver.setMetaTransactionProcessor()`
2. `ReferralValidator.updateMaxCommissionRate()`
3. `ReferralValidator.updateSigningWallet()`

Recommendation: Use the inherited `onlyAdmin()` modifier instead of adding the `require` admin check in the functions.

Update: Fixed in commit "39653998bddb039eac43b58995fd9db1086d2603".

QSP-11 Outdated Solidity Version

Severity: Informational

Status: Acknowledged

File(s) affected: `EstateSaleWithAuth.sol`, `LandToken.sol`, `ERC1155.sol`, `ERC20.sol`, `AuthValidator.sol`, `MetaTransactionReceiver.sol`, `Validator.sol`, `Admin.sol`, `SigUtil.sol`

Description: As security standards develop, so does the Solidity language. To stay up to date with current practices, it's important to use a recent version of Solidity and conventions.

Recommendation: Check the slither documentation for recommended versions of solidity.

Update: The Sandbox team commented the following: "There is WIP for a new land sale system that will take place soon (which will use a newer version of solidity)".

QSP-12 Application Monitoring Can Be Improved by Emitting More Events

Severity: Informational

Status: Fixed

File(s) affected: `EstateSaleWithAuth.sol`, `ReferralValidator.sol`

Description: In order to validate the proper deployment and initialization of the contracts, it is a good practice to emit events. Also, any important state transitions can be logged, which is beneficial for monitoring the contract, and also tracking eventual bugs or hacks. Below we present a non-exhaustive list of events that could be emitted to improve application management:

1. `EstateSaleWithAuth.setReceivingWallet()` does not emit an event reflecting changes made to the state variable `_wallet`.
2. `ReferralValidator.updateSigningWallet()` does not emit an event reflecting changes made to the state variable `_signingWallet`.
3. `ReferralValidator.updateMaxCommissionRate()` does not emit an event reflecting changes made to the state variable `_maxCommissionRate`.

Recommendation: Consider emitting the suggested events.

Update: Fixed in commit "4f05dd70858dbbf45f6870b3efcc42becfe5b41a".

QSP-13 Missing License

Severity: Informational

Status: Fixed

File(s) affected: `MetaTransactionReceiver.sol`, `LandToken.sol`, `SigUtil.sol`, `Admin.sol`, `ReferalValidator.sol`

Description: Missing `SPDX-License-Identifier` at the start of the contract. Suppose no license is wanted.

Recommendation: Add `SPDX-License-Identifier` to the suggested contracts with the appropriate license.

Update: Fixed in commit "4f05dd70858dbbf45f6870b3efcc42becfe5b41a".

QSP-14 Incorrect Use of Buyer Address in `EstateSaleWithAuth`

Severity: Undetermined

Status: Acknowledged

File(s) affected: `EstateSaleWithAuth.sol`, `ReferralValidator.sol`

Description: When submitting a transaction to buy land, the first argument is the `buyer`, which is the address the contract will call `ERC20.transferFrom(buyer, ...)` on.
The only verification for this parameter is performed in `_checkAddressesAndExpiryTime()` with the check `require(buyer == msg.sender || _metaTransactionContracts[msg.sender], "NOT_AUTHORIZED");`. This means any value is accepted as long as the transaction is submitted by an allowed meta-transaction contract.
However, it seems like using the `buyer` parameter in `ERC20.transferFrom(buyer, ...)` is incorrect, and `msg.sender` should be used instead. The following steps explain the meta-transaction processor execution:

1. As it can be found in the EIP-1776 the amount to be used as payment for the land is initially transferred to the meta transaction processor contract and then approved to the `EstateSaleWithAuth` address.
2. The external call to the `EstateSaleWithAuth` contract is executed here.
3. After the external call execution, any Sand token left will be sent back to the `buyer` here.

In conclusion, using the `buyer` as `from` address in `ReferralValidator.handleReferralWithERC20()` when calling `transferFrom()` does not make sense, since the users turned to a relayer due to unavailable ETH balance, and therefore would not have been able to approve any allowance to the receiver contract.
Secondly, it represents a risk since the `buyer` address cannot be validated from the `EstateSaleWithAuth` side, knowing that any whitelisted address in `_metaTransactionContracts` can be malicious or prone to exploits.

Recommendation: We recommend reviewing `EIP-1776` execution and possible change the `buyer` address to `msg.sender` in `ReferralValidator.handleReferralWithERC20()` when executing `ERC20.transferFrom()`.

Update: The project team acknowledged this issue as follows: "We assume that we have implemented 1776 in a specific way for our use, and not in a standard. Since we call the function from SAND.approveAndCall, and since we use EIP-2771 on the SAND contract to call the EstateSalesWithAuth contract"

Code Documentation

1. The function `ReferralValidator.getMaxCommisionRate()` contains a typo (missing an `s`).
2. TODO comments have been left in the code:
   1. `SafeMathsWithRequire:L109`.
   2. `AuthValidator:L2`.
3. Some functions are missing doc string descriptions:
   1. In `SafeMathsWithRequire`: `sqrt6()`,`sqrt3()`, `cbrt6()`, `cbrt3()`, `rt6_3()`.
   2. All functions in AuthValidator.sol.
4. In `EstateSaleWithAuth.buyLandWithSand()` some input parameters are missing documentation.

Adherence to Best Practices

1. `ReferralValidator._previousSigningDelay` can be declared as `constant` and the initialization simplified to `10 days`.
2. Commented out code can be removed in `EstateSaleWithAuth.sol:L81` .
3. Check that `admin` in `Admin.changeAdmin()` is different than the current Admin.
4. Misleading name for interface for `ERC20`, it should be called `IERC20` since it is an interface. The same remark is applicable to `ERC1155` and `LandToken` contracts.

Test Results

Test Suite Results

```
AssetERC1155:ERC1155
    bouncerAdmin
Nothing to compile
       ✓ can't set address 0 to bouncerAdmin (8579ms)
    transfers
       ✓ transferring one instance of an item results in an ERC1155 TransferSingle event (512ms)
       ✓ transferring multiple instances of an item results in an ERC1155 TransferSingle event (499ms)
       ✓ transferring zero instances of an item results in an ERC1155 TransferSingle event (434ms)
       ✓ transferring an item with 1 supply does not result in an ERC1155 TransferBatch event (720ms)
       ✓ transferring an item with >1 supply does not result in an ERC1155 TransferBatch event (470ms)
       ✓ can be transferred to a normal address (457ms)
       ✓ cannot be transferred to zero address (310ms)
       ✓ cannot transfer more items than you own (312ms)
       ✓ cannot transfer an item with supply 1 that you do not own (284ms)
       ✓ cannot transfer an item that you do not own (304ms)
       ✓ cannot transfer more item of 1 supply (311ms)
       ✓ cannot transfer to a contract that does not accept ERC1155 (371ms)
       ✓ cannot transfer multiple instances of an item to a contract that does not accept ERC1155 (355ms)
       ✓ cannot transfer an item of supply 1 to a contract that does not accept ERC1155 (573ms)
       ✓ cannot transfer an item of supply 1 to a contract that does not return the correct ERC1155_IS_RECEIVER value (293ms)
       ✓ cannot transfer an item of supply >1 to a contract that does not return the correct ERC1155_IS_RECEIVER value (297ms)
    batch transfers
       ✓ transferring an item with 1 supply results in an ERC1155 BatchTransfer event (450ms)
       ✓ transferring an item with >1 supply results in an ERC1155 BatchTransfer event (445ms)
       ✓ transferring zero items with 1 supply results in an ERC1155 BatchTransfer event (422ms)
       ✓ transferring zero items with >1 supply results in an ERC1155 BatchTransfer event (434ms)
       ✓ transferring empty list results in an ERC1155 BatchTransfer event (421ms)
       ✓ transferring multiple items results in an ERC1155 BatchTransfer event (800ms)
       ✓ transferring multiple items including zero amount results in an ERC1155 BatchTransfer event (898ms)
       ✓ transferring an item with 1 supply with batch transfer does not result in a TransferSingle event (496ms)
       ✓ transferring an item with >1 supply with batch transfer does not result in a TransferSingle event (459ms)
       ✓ can use batch transfer to send tokens to a normal address (472ms)
       ✓ cannot batch transfer the same token twice and exceed the amount owned (359ms)
       ✓ can use batch transfer to send token twice if there is sufficient amount owned (656ms)
       ✓ cannot batch transfer tokens to zeroAddress (329ms)
       ✓ cannot batch transfer tokens if array lengths do not match (366ms)
       ✓ cannot batch transfer more than the amount owned (555ms)
       ✓ cannot batch transfer more items of 1 supply (307ms)
       ✓ cannot batch transfer to a contract that does not accept ERC1155 (317ms)
       ✓ cannot batch transfer to a contract that does not return the correct magic value (322ms)
       ✓ can batch transfer to a contract that does accept ERC1155 and which returns the correct magic value (748ms)
       ✓ can batch transfer item with 1 or more supply at the same time (603ms)
       ✓ can obtain balance of batch (556ms)
    approvalForAll
       ✓ setting approval results in ApprovalForAll event (340ms)
       ✓ setting approval fails if sender is operator (493ms)
       ✓ operator cannot transfer without approval (299ms)
       ✓ operator can transfer after approval (456ms)
       ✓ operator cannot transfer after approval is removed (581ms)
    supportsInterface
       ✓ contract claims to supports ERC165 (285ms)
       ✓ contract does not claim to support random interface (263ms)
       ✓ contract does not claim to support invalid interface (271ms)
    ordering
       ✓ transfer empty array (359ms)
       ✓ transfer multiple items in any order (i) (628ms)
       ✓ transfer multiple items in any order (ii) (613ms)
       ✓ transfer multiple items in any order (iii) (751ms)
       ✓ transfer multiple items in any order (iv) (761ms)
       ✓ transfer multiple items in any order (v) (584ms)
       ✓ transfer multiple items in any order (vi) (661ms)
       ✓ transfer multiple items in any order (vii) (663ms)
       ✓ transfer multiple items in any order (viii) (672ms)
       ✓ transfer multiple items in any order twice (i) (920ms)
       ✓ transfer multiple items in any order twice (ii) (1118ms)
       ✓ transfer multiple items in any order twice (iii) (862ms)
       ✓ transfer multiple items in any order twice (iv) (1031ms)
       ✓ transfer multiple items in any order twice (v) (1100ms)
       ✓ transfer multiple items in any order twice (vi) (922ms)

  Asset:ERC721
    non existing NFT
       ✓ transfering a non existing NFT fails (1106ms)
       ✓ tx balanceOf a zero owner fails (53ms)
       ✓ call balanceOf a zero owner fails (62ms)
```

```
      ✓ tx ownerOf a non existing NFT fails (50ms)
      ✓ call ownerOf a non existing NFT fails (58ms)
      ✓ tx getApproved a non existing NFT fails (48ms)
      ✓ call getApproved a non existing NFT fails (60ms)
    balance
      ✓ balance is zero for new user (54ms)
      ✓ balance return correct value (283ms)
    mint
      ✓ mint result in a transfer from 0 event (67ms)
      ✓ mint for gives correct owner (71ms)
    burnAsset
      ✓ burn result in a transfer to 0 event (125ms)
      ✓ burn result in ownerOf throwing (138ms)
    transfer
      ✓ transfering one NFT results in one erc721 transfer event (118ms)
      ✓ transfering one NFT change to correct owner (131ms)
      ✓ transfering one NFT increase new owner balance (129ms)
      ✓ transfering one NFT decrease past owner balance (131ms)
      ✓ transfering from without approval should fails (58ms)
      ✓ transfering to zero address should fails (50ms)
      ✓ transfering to a contract that do not accept erc721 token should not fail (133ms)
    safeTransfer
      ✓ safe transfering one NFT results in one erc721 transfer event (125ms)
      ✓ safe transfering to zero address should fails (53ms)
      ✓ safe transfering one NFT change to correct owner (130ms)
      ✓ safe transfering from without approval should fails (47ms)
      ✓ safe transfering to a contract that do not accept erc721 token should fail (72ms)
      ✓ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (70ms)
      ✓ safe transfering to a contract that do not implemented onERC721Received should fail (69ms)
      ✓ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (169ms)
    safeTransfer with empty bytes
      ✓ data:0x : safe transfering one NFT results in one erc721 transfer event (128ms)
      ✓ data:0x : safe transfering to zero address should fails (48ms)
      ✓ data:0x : safe transfering one NFT change to correct owner (133ms)
      ✓ data:0x : safe transfering from without approval should fails (48ms)
      ✓ data:0x : safe transfering to a contract that do not accept erc721 token should fail (68ms)
      ✓ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (67ms)
      ✓ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (66ms)
      ✓ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (169ms)
    safeTransfer with data
      ✓ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (121ms)
      ✓ data:0xff56fe3422 : safe transfering to zero address should fails (56ms)
      ✓ data:0xff56fe3422 : safe transfering one NFT change to correct owner (131ms)
      ✓ data:0xff56fe3422 : safe transfering from without approval should fails (50ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (70ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (71ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (71ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (170ms)
    ERC165
      ✓ claim to support erc165 (57ms)
      ✓ claim to support base erc721 interface (48ms)
      ✓ claim to support erc721 metadata interface (52ms)
      ✓ does not claim to support random interface (44ms)
      ✓ does not claim to support the invalid interface (50ms)
    Approval
      ✓ approving emit Approval event (89ms)
      ✓ removing approval emit Approval event (139ms)
      ✓ approving update the approval status (94ms)
      ✓ cant approve if not owner or operator  (117ms)
      ✓ approving allows transfer from the approved party (174ms)
      ✓ transfering the approved NFT results in aproval reset for it (169ms)
      ✓ transfering the approved NFT again will fail (179ms)
      ✓ approval by operator works (242ms)
    ApprovalForAll
      ✓ approving all emit ApprovalForAll event (90ms)
      ✓ approving all update the approval status (95ms)
      ✓ unsetting approval for all should update the approval status (130ms)
      ✓ unsetting approval for all should emit ApprovalForAll event (131ms)
      ✓ approving for all allows transfer from the approved party (181ms)
      ✓ transfering one NFT do not results in aprovalForAll reset (168ms)
      ✓ approval for all does not grant approval on a transfered NFT (166ms)
      ✓ approval for all set before will work on a transfered NFT (249ms)
      ✓ approval for all allow to set individual nft approve (275ms)

  PolygonAssetERC1155:ERC1155
    bouncerAdmin
      ✓ can't set address 0 to bouncerAdmin (2242ms)
    transfers
      ✓ transferring one instance of an item results in an ERC1155 TransferSingle event (394ms)
      ✓ transferring multiple instances of an item results in an ERC1155 TransferSingle event (378ms)
      ✓ transferring zero instances of an item results in an ERC1155 TransferSingle event (298ms)
      ✓ transferring an item with 1 supply does not result in an ERC1155 TransferBatch event (368ms)
      ✓ transferring an item with >1 supply does not result in an ERC1155 TransferBatch event (576ms)
      ✓ can be transferred to a normal address (373ms)
      ✓ cannot be transferred to zero address (253ms)
      ✓ cannot transfer more items than you own (259ms)
      ✓ cannot transfer an item with supply 1 that you do not own (252ms)
      ✓ cannot transfer an item that you do not own (240ms)
      ✓ cannot transfer more item of 1 supply (249ms)
      ✓ cannot transfer to a contract that does not accept ERC1155 (247ms)
      ✓ cannot transfer multiple instances of an item to a contract that does not accept ERC1155 (253ms)
      ✓ cannot transfer an item of supply 1 to a contract that does not accept ERC1155 (251ms)
      ✓ cannot transfer an item of supply 1 to a contract that does not return the correct ERC1155_IS_RECEIVER value (254ms)
      ✓ cannot transfer an item of supply >1 to a contract that does not return the correct ERC1155_IS_RECEIVER value (270ms)
    batch transfers
      ✓ transferring an item with 1 supply results in an ERC1155 BatchTransfer event (453ms)
      ✓ transferring an item with >1 supply results in an ERC1155 BatchTransfer event (658ms)
      ✓ transferring zero items with 1 supply results in an ERC1155 BatchTransfer event (338ms)
      ✓ transferring zero items with >1 supply results in an ERC1155 BatchTransfer event (332ms)
      ✓ transferring empty list results in an ERC1155 BatchTransfer event (371ms)
      ✓ transferring multiple items results in an ERC1155 BatchTransfer event (790ms)
      ✓ transferring multiple items including zero amount results in an ERC1155 BatchTransfer event (591ms)
      ✓ transferring an item with 1 supply with batch transfer does not result in a TransferSingle event (448ms)
      ✓ transferring an item with >1 supply with batch transfer does not result in a TransferSingle event (431ms)
      ✓ can use batch transfer to send tokens to a normal address (602ms)
      ✓ cannot batch transfer the same token twice and exceed the amount owned (299ms)
      ✓ can use batch transfer to send token twice if there is sufficient amount owned (570ms)
      ✓ cannot batch transfer tokens to zeroAddress (258ms)
      ✓ cannot batch transfer tokens if array lengths do not match (248ms)
      ✓ cannot batch transfer more than the amount owned (243ms)
      ✓ cannot batch transfer more items of 1 supply (247ms)
      ✓ cannot batch transfer to a contract that does not accept ERC1155 (284ms)
      ✓ cannot batch transfer to a contract that does not return the correct magic value (282ms)
      ✓ can batch transfer to a contract that does accept ERC1155 and which returns the correct magic value (622ms)
      ✓ can batch transfer item with 1 or more supply at the same time (484ms)
      ✓ can obtain balance of batch (663ms)
    approvalForAll
      ✓ setting approval results in ApprovalForAll event (296ms)
      ✓ setting approval fails if sender is operator (241ms)
      ✓ operator cannot transfer without approval (259ms)
      ✓ operator can transfer after approval (469ms)
      ✓ operator cannot transfer after approval is removed (559ms)
    supportsInterface
      ✓ contract claims to supports ERC165 (271ms)
      ✓ contract does not claim to support random interface (288ms)
      ✓ contract does not claim to support invalid interface (248ms)
    ordering
      ✓ transfer empty array (341ms)
      ✓ transfer multiple items in any order (i) (2368ms)
      ✓ transfer multiple items in any order (ii) (839ms)
      ✓ transfer multiple items in any order (iii) (568ms)
      ✓ transfer multiple items in any order (iv) (707ms)
      ✓ transfer multiple items in any order (v) (536ms)
      ✓ transfer multiple items in any order (vi) (659ms)
      ✓ transfer multiple items in any order (vii) (688ms)
      ✓ transfer multiple items in any order (viii) (811ms)
      ✓ transfer multiple items in any order twice (i) (739ms)
      ✓ transfer multiple items in any order twice (ii) (1069ms)
      ✓ transfer multiple items in any order twice (iii) (840ms)
      ✓ transfer multiple items in any order twice (iv) (1094ms)
      ✓ transfer multiple items in any order twice (v) (1129ms)
      ✓ transfer multiple items in any order twice (vi) (940ms)

  GameToken:ERC721
    non existing NFT
      ✓ transfering a non existing NFT fails (3452ms)
      ✓ tx balanceOf a zero owner fails (82ms)
      ✓ call balanceOf a zero owner fails (93ms)
      ✓ tx ownerOf a non existing NFT fails (92ms)
      ✓ call ownerOf a non existing NFT fails (100ms)
      ✓ tx getApproved a non existing NFT fails (82ms)
      ✓ call getApproved a non existing NFT fails (81ms)
    balance
      ✓ balance is zero for new user (87ms)
      ✓ balance return correct value (349ms)
    mint
      ✓ mint result in a transfer from 0 event (107ms)
      ✓ mint for gives correct owner (110ms)
    burn
      ✓ burn result in a transfer to 0 event (170ms)
      ✓ burn result in ownerOf throwing (187ms)
    batchTransfer
      ✓ batch transfer of same NFT ids should fails (91ms)
      ✓ batch transfer works (211ms)
    mandatory batchTransfer
      ✓ batch transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (220ms)
      ✓ batch transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (105ms)
      ✓ batch transfering to a contract that do not accept erc721 token should fail (112ms)
      ✓ batch transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (101ms)
      ✓ batch transfering to a contract that do not implemented mandatory receiver should not fail (203ms)
      ✓ batch transfering to a contract that return the correct onERC721Received bytes shoudl succeed (242ms)
    mandatory transfer
      ✓ transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (194ms)
      ✓ transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (101ms)
      ✓ transfering to a contract that do not accept erc721 token should fail (101ms)
      ✓ transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (102ms)
      ✓ transfering to a contract that do not implemented mandatory receiver should not fail (189ms)
      ✓ transfering to a contract that return the correct onERC721Received bytes shoudl succeed (246ms)
    safe batch transfer
      ✓ safe batch transfer of same NFT ids should fails (94ms)
      ✓ safe batch transfer works (211ms)
      ✓ safe batch transfering to a contract that do not implemented onERC721Received should fail (106ms)
      ✓ safe batch transfering to a contract that implements onERC721Received should succeed (326ms)
    transfer
      ✓ transfering one NFT results in one erc721 transfer event (165ms)
      ✓ transfering one NFT change to correct owner (168ms)
      ✓ transfering one NFT increase new owner balance (191ms)
      ✓ transfering one NFT decrease past owner balance (172ms)
      ✓ transfering from without approval should fails (88ms)
      ✓ transfering to zero address should fails (88ms)
      ✓ transfering to a contract that do not accept erc721 token should not fail (196ms)
    safeTransfer
      ✓ safe transfering one NFT results in one erc721 transfer event (157ms)
      ✓ safe transfering to zero address should fails (91ms)
      ✓ safe transfering one NFT change to correct owner (160ms)
      ✓ safe transfering from without approval should fails (82ms)
      ✓ safe transfering to a contract that do not accept erc721 token should fail (96ms)
      ✓ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (114ms)
      ✓ safe transfering to a contract that do not implemented onERC721Received should fail (106ms)
      ✓ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (217ms)
    safeTransfer with empty bytes
      ✓ data:0x : safe transfering one NFT results in one erc721 transfer event (159ms)
      ✓ data:0x : safe transfering to zero address should fails (89ms)
      ✓ data:0x : safe transfering one NFT change to correct owner (163ms)
      ✓ data:0x : safe transfering from without approval should fails (85ms)
      ✓ data:0x : safe transfering to a contract that do not accept erc721 token should fail (101ms)
      ✓ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (95ms)
      ✓ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (94ms)
      ✓ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (207ms)
    safeTransfer with data
      ✓ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (154ms)
      ✓ data:0xff56fe3422 : safe transfering to zero address should fails (82ms)
      ✓ data:0xff56fe3422 : safe transfering one NFT change to correct owner (174ms)
      ✓ data:0xff56fe3422 : safe transfering from without approval should fails (90ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (102ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (111ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (101ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (210ms)
    ERC165
      ✓ claim to support erc165 (85ms)
      ✓ claim to support base erc721 interface (76ms)
      ✓ claim to support erc721 metadata interface (75ms)
      ✓ does not claim to support random interface (86ms)
      ✓ does not claim to support the invalid interface (84ms)
    Approval
      ✓ approving emit Approval event (133ms)
      ✓ removing approval emit Approval event (225ms)
      ✓ approving update the approval status (146ms)
      ✓ cant approve if not owner or operator  (152ms)
      ✓ approving allows transfer from the approved party (224ms)
      ✓ transfering the approved NFT results in aproval reset for it (218ms)
      ✓ transfering the approved NFT again will fail (227ms)
      ✓ approval by operator works (283ms)
    ApprovalForAll
      ✓ approving all emit ApprovalForAll event (130ms)
      ✓ approving all update the approval status (127ms)
      ✓ unsetting approval for all should update the approval status (184ms)
      ✓ unsetting approval for all should emit ApprovalForAll event (195ms)
      ✓ approving for all allows transfer from the approved party (212ms)
      ✓ transfering one NFT do not results in aprovalForAll reset (201ms)
      ✓ approval for all does not grant approval on a transfered NFT (200ms)
      ✓ approval for all set before will work on a transfered NFT (276ms)
      ✓ approval for all allow to set individual nft approve (369ms)

  LandBaseToken:ERC721
    non existing NFT
      ✓ transfering a non existing NFT fails (2616ms)
      ✓ tx balanceOf a zero owner fails (38ms)
      ✓ call balanceOf a zero owner fails
      ✓ tx ownerOf a non existing NFT fails (38ms)
      ✓ call ownerOf a non existing NFT fails
      ✓ tx getApproved a non existing NFT fails (39ms)
      ✓ call getApproved a non existing NFT fails
    balance
      ✓ balance is zero for new user
      ✓ balance return correct value (215ms)
    mint
      ✓ mint result in a transfer from 0 event (39ms)
      ✓ mint for gives correct owner (43ms)
    burn
      ✓ burn result in a transfer to 0 event (87ms)
      ✓ burn result in ownerOf throwing (91ms)
    batchTransfer
      ✓ batch transfer of same NFT ids should fails (39ms)
      ✓ batch transfer works (115ms)
    mandatory batchTransfer
      ✓ batch transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (125ms)
      ✓ batch transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (51ms)
      ✓ batch transfering to a contract that do not accept erc721 token should fail (51ms)
      ✓ batch transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (61ms)
      ✓ batch transfering to a contract that do not implemented mandatory receiver should not fail (126ms)
      ✓ batch transfering to a contract that return the correct onERC721Received bytes shoudl succeed (157ms)
    mandatory transfer
      ✓ transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (117ms)
      ✓ transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (48ms)
      ✓ transfering to a contract that do not accept erc721 token should fail (47ms)
      ✓ transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (54ms)
      ✓ transfering to a contract that do not implemented mandatory receiver should not fail (110ms)
      ✓ transfering to a contract that return the correct onERC721Received bytes shoudl succeed (133ms)
    safe batch transfer
      ✓ safe batch transfer of same NFT ids should fails
      ✓ safe batch transfer works (111ms)
      ✓ safe batch transfering to a contract that do not implemented onERC721Received should fail (47ms)
      ✓ safe batch transfering to a contract that implements onERC721Received should succeed (193ms)
    transfer
      ✓ transfering one NFT results in one erc721 transfer event (83ms)
      ✓ transfering one NFT change to correct owner (90ms)
      ✓ transfering one NFT increase new owner balance (88ms)
      ✓ transfering one NFT decrease past owner balance (89ms)
      ✓ transfering from without approval should fails (45ms)
      ✓ transfering to zero address should fails
      ✓ transfering to a contract that do not accept erc721 token should not fail (101ms)
    safeTransfer
      ✓ safe transfering one NFT results in one erc721 transfer event (87ms)
      ✓ safe transfering to zero address should fails
      ✓ safe transfering one NFT change to correct owner (89ms)
      ✓ safe transfering from without approval should fails
      ✓ safe transfering to a contract that do not accept erc721 token should fail (47ms)
      ✓ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (56ms)
      ✓ safe transfering to a contract that do not implemented onERC721Received should fail (47ms)
      ✓ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (125ms)
    safeTransfer with empty bytes
      ✓ data:0x : safe transfering one NFT results in one erc721 transfer event (93ms)
      ✓ data:0x : safe transfering to zero address should fails
      ✓ data:0x : safe transfering one NFT change to correct owner (94ms)
      ✓ data:0x : safe transfering from without approval should fails
      ✓ data:0x : safe transfering to a contract that do not accept erc721 token should fail (46ms)
      ✓ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (45ms)
      ✓ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (55ms)
      ✓ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (128ms)
    safeTransfer with data
      ✓ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (94ms)
      ✓ data:0xff56fe3422 : safe transfering to zero address should fails (40ms)
      ✓ data:0xff56fe3422 : safe transfering one NFT change to correct owner (100ms)
      ✓ data:0xff56fe3422 : safe transfering from without approval should fails
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (46ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (46ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (53ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (125ms)
    ERC165
      ✓ claim to support erc165
      ✓ claim to support base erc721 interface
      ✓ claim to support erc721 metadata interface
      ✓ does not claim to support random interface
      ✓ does not claim to support the invalid interface
    Approval
      ✓ approving emit Approval event (79ms)
      ✓ removing approval emit Approval event (122ms)
      ✓ approving update the approval status (83ms)
      ✓ cant approve if not owner or operator  (93ms)
      ✓ approving allows transfer from the approved party (146ms)
      ✓ transfering the approved NFT results in aproval reset for it (141ms)
      ✓ transfering the approved NFT again will fail (136ms)
      ✓ approval by operator works (199ms)
    ApprovalForAll
      ✓ approving all emit ApprovalForAll event (68ms)
      ✓ approving all update the approval status (83ms)
      ✓ unsetting approval for all should update the approval status (121ms)
      ✓ unsetting approval for all should emit ApprovalForAll event (120ms)
      ✓ approving for all allows transfer from the approved party (156ms)
      ✓ transfering one NFT do not results in aprovalForAll reset (138ms)
      ✓ approval for all does not grant approval on a transfered NFT (147ms)
      ✓ approval for all set before will work on a transfered NFT (199ms)
      ✓ approval for all allow to set individual nft approve (246ms)

  SafeMathWithRequire
{ loopCounter: 81 }
{ loopCounter: 173 }
    ✓ cbrt6
{ loopCounter: 47 }
{ loopCounter: 139 }
    ✓ cbrt3
{ loopCounter: 215 }
{ loopCounter: 469 }
    ✓ rt6_3 (53ms)

  LandWeightedSANDRewardPool computation
    ✓ computing contributions (3615ms)

  MockSANDRewardPool
    ✓ Pool contains reward tokens (367ms)
    ✓ User with stakeTokens can stake (38ms)
    ✓ User earnings for 0 NFTs match expected reward (39ms)
    ✓ User earnings for 0 NFTs match expected reward with 1 stake (38ms)
    ✓ User earnings for 0 NFTs match expected reward with 2 stakes (62ms)
    ✓ User earnings for 0 NFTs match expected reward with 3 stakes (104ms)
    ✓ User earnings for 0 NFTs match expected reward with 4 stakes (130ms)
    ✓ User earnings for 0 NFTs match expected reward with 10 stakes (314ms)
    ✓ User earnings for 1 NFTs match expected reward (78ms)
    ✓ User earnings for 1 NFTs match expected reward with 10 stakes (593ms)
    ✓ User earnings for 2 NFTs match expected reward (86ms)
    ✓ User earnings for 3 NFTs match expected reward (94ms)
    ✓ User earnings for 3 NFTs match expected reward with 10 stakes (676ms)
    ✓ User earnings for 89 NFTs match expected reward (868ms)
    ✓ User earnings for 89 NFTs match expected reward with 10 stakes (1525ms)
    ✓ Multiple Users' earnings for 0 NFTs match expected reward: 2 users (84ms)
    ✓ Multiple Users' earnings for 0 NFTs match expected reward: 2 users, 10 stakes each (625ms)
    ✓ Multiple Users' earnings for 0 NFTs match expected reward: 3 users, 1 stake each (137ms)
    ✓ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 1 stake each (148ms)
    ✓ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 10 stakes each (1154ms)
    ✓ Multiple Users' earnings for 3 NFTs match expected reward: 2 users, 1 stake each (206ms)
    ✓ Multiple Users' earnings for 100 NFTs match expected reward: 2 users, 1 stake each (1949ms)
    ✓ Staking with STAKE_AMOUNT plus an extra amount equivalent to 2 NFTs (54ms)
    ✓ Earlier staker gets more rewards with same NFT amount - small NFT number (140ms)
    ✓ Earlier staker gets more rewards with same NFT amount - large NFT number (1897ms)
    ✓ More lands give more rewards than earlier staker when NFT amounts are smaller (160ms)
    ✓ More lands do not give more rewards than earlier staker with large NFT amounts (1926ms)
    ✓ rewardToken in pool is more than amount notified (527ms)
    ✓ rewardToken in pool is zero (338ms)
    ✓ rewardToken in pool is less than amount notified (68ms)
    ✓ the call to notifyRewardAmount is made after users first call stake (340ms)
    ✓ user is earning rewards and pool is notified for a second time before end of current reward period (363ms)

  ActualSANDRewardPool
    ✓ Contract should exist (2985ms)
    ✓ Pool contains reward tokens
    ✓ User with stakeTokens can stake (39ms)
    ✓ User can earn rewardTokens if pool has been notified of reward (49ms)
    ✓ admin can notifyRewardAmount and start a new reward process (without sending more reward tokens)
    ✓ User cannot earn rewardTokens if they stake after the end time (41ms)
    ✓ User earns full reward amount if they are the only staker after 1 day (39ms)
    ✓ User earns full reward amount if they are the only staker after 29 days (44ms)
    ✓ User with 0 LAND earns correct reward amount (54ms)
    ✓ User with 0 LAND earns correct reward amount - smaller stake (47ms)
    ✓ User with 1 LAND earns correct reward amount (89ms)
    ✓ User with 3 LANDs earns correct reward amount (126ms)
    ✓ User with 10 LANDs earns correct reward amount (204ms)
    ✓ User can withdraw some stakeTokens after several amounts have been staked (92ms)
    ✓ First user can withdraw their stakeTokens (68ms)
    ✓ User can withdraw all stakeTokens after several amounts have been staked (126ms)
    ✓ First user can claim their reward - no NFTs (67ms)
    ✓ First user can claim their reward - has NFTs (195ms)
    ✓ A user can claim their reward after multiple stakes (348ms)
    ✓ First user can exit the pool (68ms)
    ✓ A user can exit the pool after multiple stakes (138ms)
    ✓ A user with NFTs can exit the pool after multiple stakes (345ms)

  PolygonCatalyst_EPIC
    ✓ transfering from users[0] to users[1] should adjust their balance accordingly (2876ms)
    ✓ transfering from users[0] more token that it owns should fails
    ✓ transfering to address zero should fails
    ✓ transfering to address(this) should fail
    ✓ transfering from users[0] to users[1] by users[0] should adjust their balance accordingly (86ms)
    ✓ transfering from users[0] by users[1] should fails
    ✓ 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 (156ms)
    ✓ transfering from users[0] to users[1] by operator after approval and approval reset, should fail (126ms)
    ✓ transfering from users[0] to users[1] by operator after approval, should adjust the operator alowance accordingly (167ms)
    ✓ transfering from users[0] to users[1] by operator after max approval (2**256-1), should NOT adjust the operator allowance (164ms)
    ✓ transfering from users[0] to users[1] by operator after approval, but without enough allowance, should fails (85ms)
    ✓ transfering from users[0] by operators without pre-approval should fails
    ✓ approving operator should trigger a Approval event (66ms)
    ✓ disapproving operator (allowance to zero) should trigger a Approval event (129ms)
    ✓ approve to address zero should fails

  PolygonGem_POWER
    ✓ transfering from users[0] to users[1] should adjust their balance accordingly (3076ms)
    ✓ transfering from users[0] more token that it owns should fails
    ✓ transfering to address zero should fails
    ✓ transfering to address(this) should fail
    ✓ transfering from users[0] to users[1] by users[0] should adjust their balance accordingly (85ms)
    ✓ transfering from users[0] by users[1] should fails
    ✓ transfering from users[0] to users[1] should trigger a transfer event (75ms)
    ✓ transfering from users[0] to users[1] by operator after approval, should adjust their balance accordingly (167ms)
    ✓ transfering from users[0] to users[1] by operator after approval and approval reset, should fail (124ms)
    ✓ 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 (149ms)
    ✓ transfering from users[0] to users[1] by operator after approval, but without enough allowance, should fails (71ms)
    ✓ transfering from users[0] by operators without pre-approval should fails
    ✓ approving operator should trigger a Approval event (63ms)
    ✓ disapproving operator (allowance to zero) should trigger a Approval event (125ms)
    ✓ approve to address zero should fails

  PolygonLandBaseToken:ERC721
    non existing NFT
      ✓ transfering a non existing NFT fails (879ms)
      ✓ tx balanceOf a zero owner fails (71ms)
      ✓ call balanceOf a zero owner fails (74ms)
      ✓ tx ownerOf a non existing NFT fails (87ms)
      ✓ call ownerOf a non existing NFT fails (75ms)
      ✓ tx getApproved a non existing NFT fails (73ms)
      ✓ call getApproved a non existing NFT fails (81ms)
    balance
      ✓ balance is zero for new user (68ms)
      ✓ balance return correct value (161ms)
    mint
      ✓ mint result in a transfer from 0 event (85ms)
      ✓ mint for gives correct owner (86ms)
    burn
      ✓ burn result in a transfer to 0 event (145ms)
      ✓ burn result in ownerOf throwing (158ms)
    batchTransfer
      ✓ batch transfer of same NFT ids should fails (75ms)
      ✓ batch transfer works (209ms)
    mandatory batchTransfer
      ✓ batch transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (108ms)
      ✓ batch transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (100ms)
      ✓ batch transfering to a contract that do not accept erc721 token should fail (95ms)
      ✓ batch transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (92ms)
      ✓ batch transfering to a contract that do not implemented mandatory receiver should not fail (110ms)
      ✓ batch transfering to a contract that return the correct onERC721Received bytes shoudl succeed (123ms)
    mandatory transfer
      ✓ transfering to a contract that do not implements mandatory erc721 receiver but implement classic ERC721 receiver and reject should not fails (192ms)
      ✓ transfering to a contract that implements mandatory erc721 receiver (and signal it properly via 165) should fails if it reject it (109ms)
      ✓ transfering to a contract that do not accept erc721 token should fail (84ms)
      ✓ transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (91ms)
      ✓ transfering to a contract that do not implemented mandatory receiver should not fail (172ms)
      ✓ transfering to a contract that return the correct onERC721Received bytes shoudl succeed (120ms)
    safe batch transfer
      ✓ safe batch transfer of same NFT ids should fails (74ms)
      ✓ safe batch transfer works (227ms)
      ✓ safe batch transfering to a contract that do not implemented onERC721Received should fail (97ms)
      ✓ safe batch transfering to a contract that implements onERC721Received should succeed (354ms)
    transfer
      ✓ transfering one NFT results in one erc721 transfer event (90ms)
      ✓ transfering one NFT change to correct owner (97ms)
      ✓ transfering one NFT increase new owner balance (98ms)
      ✓ transfering one NFT decrease past owner balance (98ms)
      ✓ transfering from without approval should fails (70ms)
      ✓ transfering to zero address should fails (69ms)
      ✓ transfering to a contract that do not accept erc721 token should not fail (182ms)
    safeTransfer
      ✓ safe transfering one NFT results in one erc721 transfer event (91ms)
      ✓ safe transfering to zero address should fails (69ms)
      ✓ safe transfering one NFT change to correct owner (94ms)
      ✓ safe transfering from without approval should fails (71ms)
      ✓ safe transfering to a contract that do not accept erc721 token should fail (89ms)
      ✓ safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (92ms)
      ✓ safe transfering to a contract that do not implemented onERC721Received should fail (80ms)
      ✓ safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (118ms)
    safeTransfer with empty bytes
      ✓ data:0x : safe transfering one NFT results in one erc721 transfer event (94ms)
      ✓ data:0x : safe transfering to zero address should fails (67ms)
      ✓ data:0x : safe transfering one NFT change to correct owner (94ms)
      ✓ data:0x : safe transfering from without approval should fails (69ms)
      ✓ data:0x : safe transfering to a contract that do not accept erc721 token should fail (89ms)
      ✓ data:0x : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (89ms)
      ✓ data:0x : safe transfering to a contract that do not implemented onERC721Received should fail (83ms)
      ✓ data:0x : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (112ms)
    safeTransfer with data
      ✓ data:0xff56fe3422 : safe transfering one NFT results in one erc721 transfer event (102ms)
      ✓ data:0xff56fe3422 : safe transfering to zero address should fails (70ms)
      ✓ data:0xff56fe3422 : safe transfering one NFT change to correct owner (98ms)
      ✓ data:0xff56fe3422 : safe transfering from without approval should fails (68ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not accept erc721 token should fail (79ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not return the correct onERC721Received bytes shoudl fail (90ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that do not implemented onERC721Received should fail (92ms)
      ✓ data:0xff56fe3422 : safe transfering to a contract that return the correct onERC721Received bytes shoudl succeed (122ms)
    ERC165
      ✓ claim to support erc165 (68ms)
      ✓ claim to support base erc721 interface (68ms)
      ✓ claim to support erc721 metadata interface (67ms)
      ✓ does not claim to support random interface (67ms)
      ✓ does not claim to support the invalid interface (68ms)
    Approval
      ✓ approving emit Approval event (89ms)
      ✓ removing approval emit Approval event (104ms)
      ✓ approving update the approval status (92ms)
      ✓ cant approve if not owner or operator  (97ms)
      ✓ approving allows transfer from the approved party (124ms)
      ✓ transfering the approved NFT results in aproval reset for it (122ms)
      ✓ transfering the approved NFT again will fail (119ms)
      ✓ approval by operator works (190ms)
    ApprovalForAll
      ✓ approving all emit ApprovalForAll event (88ms)
      ✓ approving all update the approval status (84ms)
      ✓ unsetting approval for all should update the approval status (141ms)
      ✓ unsetting approval for all should emit ApprovalForAll event (147ms)
      ✓ approving for all allows transfer from the approved party (110ms)
      ✓ transfering one NFT do not results in aprovalForAll reset (109ms)
      ✓ approval for all does not grant approval on a transfered NFT (107ms)
      ✓ approval for all set before will work on a transfered NFT (183ms)
```

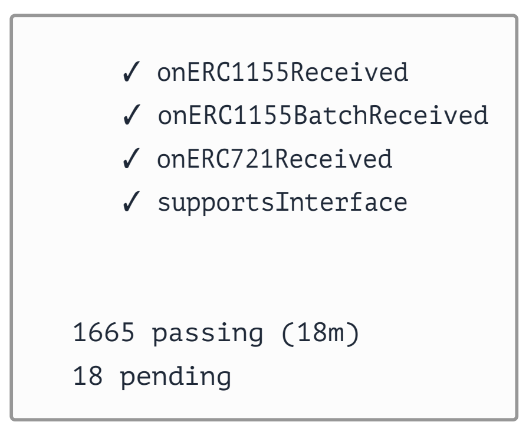
```
        ✓ approval for all allow to set individual nft approve (224ms)

  AssetERC1155.sol
20
      ✓ user sending asset to itself keep the same balance (4318ms)
      ✓ mintMultiple reverts when ids and amounts length mismatch
      ✓ can transfer assets (113ms)
      ✓ user batch sending asset to itself keep the same balance (114ms)
      ✓ user batch sending in series whose total is more than its balance (111ms)
      ✓ user batch sending more asset that it owns should fails (101ms)
      ✓ can get the chainIndex from the tokenId, supply > 1 (94ms)
      ✓ can get the chainIndex from the tokenId, supply 1 (85ms)
      ✓ can get the URI for an asset of amount 1 (136ms)
      ✓ can get the URI for a FT (133ms)
      ✓ fails get the URI for an invalid tokeId
    AssetERC1155: MetaTransactions
        ✓ can transfer by metaTx (140ms)
META TX CALL Error: OPERATOR_!AUTH
        ✓ fails to transfer someone else token by metaTx (132ms)
        ✓ can batch-transfer by metaTx (232ms)

  AssetERC721.sol
    initialization
        ✓ creation (2762ms)
        ✓ interfaces
    roles
        ✓ Can mint without metadata (although this is not the expected implementation) (179ms)
        ✓ metaTX trusted forwarder (72ms)
      admin
          ✓ admin role is set
          ✓ admin can set the trusted forwarder
          ✓ other should fail to set the trusted forwarder (50ms)
      minter
          ✓ mint with metadata (152ms)

  assetSignedAuctionWithAuth
setting AssetSignedAuction as super operator for Asset
setting AssetSignedAuction as super operator for Sand
setting AssetSignedAuctionWithAuth as super operator for Asset
setting AssetSignedAuctionWithAuth as super operator for Sand
      ✓ should be able to set fee (4730ms)
      ✓ should be able to set feeLimit
      ✓ should fail setting feeLimit - no admin
      ✓ should fail setting feeLimit - above max limit
      ✓ should fail setting fee - no admin
      ✓ should fail is buyer == seller (134ms)
      ✓ should fail is ids.length != amounts.length (131ms)
      ✓ should fail - insuficient amount (121ms)
      ✓ should be able to claim seller offer in ETH (170ms)
      ✓ should NOT be able to claim offer if signature mismatches (131ms)
      ✓ should NOT be able to claim offer with invalid backend signature (117ms)
      ✓ should be able to claim seller offer in SAND (189ms)
      ✓ should be able to cancel offer (105ms)
      ✓ should NOT be able to claim offer without enough SAND (158ms)
      ✓ should NOT be able to claim offer if it did not start yet (118ms)
      ✓ should NOT be able to claim offer if it already ended (114ms)

  Test first Asset contract and upgrade process for splitting into ERC1155 and ERC721
    transfer
        ✓ user sending asset to itself keep the same balance (1118ms)
        ✓ can transfer assets
        ✓ user batch sending asset to itself keep the same balance
        ✓ user batch sending in series whose total is more than its balance
        ✓ user batch sending more asset that it owns should fails
    bitwise operations, collections and extraction
        ✓ cannot get the chainIndex from the tokenId when supply > 1
        ✓ cannot get the chainIndex from the tokenId when supply == 1
        ✓ collectionIndexOf reverts if provided ID is not an ERC721 and supply > 1
        ✓ collectionIndexOf reverts if provided ID is not an ERC721 and supply == 1
        ✓ user cannot extract an ERC1155 if supply == 1
      - user can extract an ERC1155 if supply > 1
      - user can extract an ERC1155 if supply == 2
      - user cannot extract an ERC1155 if supply == 2 but 1 has already been extracted
      - user can extract ERC721 more than once
    tokenURI
        ✓ can get the URI for an asset of amount 1 (39ms)
        ✓ can get the URI for a FT
        ✓ fails get the URI for an invalid tokeId
    Upgrade and split Asset into ERC1155 and ERC1155
        ✓ should upgrade to AssetERC1155 and keep storage intact (1192ms)

  Asset_Giveaway
      ✓ User cannot claim when test contract holds zero assets (167ms)
      ✓ User can claim allocated multiple assets for multiple assetIds from Giveaway contract (321ms)
      ✓ Claimed Event is emitted for successful claim (39ms)
      ✓ User can claim allocated single asset for single assetId from Giveaway contract
      ✓ User tries to claim the wrong amount of an assetID
      ✓ User cannot claim their assets more than once (43ms)
      ✓ 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 (305ms)
      ✓ merkleRoot cannot be set twice (104ms)
      ✓ merkleRoot can only be set by admin

  GAS:Asset_Giveaway_1:Claiming
      ✓ 1 claim (179ms)
      ✓ 10 claims (162ms)
      ✓ 4000 claims (1292ms)
      ✓ 10000 claims (2586ms)
{
  "Gas per claim - 1 claim total": 111935,
  "Gas per claim - 10 claims total": 115734,
  "Gas per claim - 4000 claims total": 125818,
  "Gas per claim - 10000 claims total": 128352
}

  MerkleTree_assets
      ✓ should validate the data

  SignedGiveaway.sol
    initialization
        ✓ interfaces (817ms)
    roles
        ✓ admin
        ✓ signer
    claim
        ✓ should be able to claim sand (43ms)
        ✓ should fail to claim the same id twice (49ms)
        ✓ should fail to claim if the signature is wrong
        ✓ should fail to mint if the signer is invalid
        ✓ claim with metaTX trusted forwarder (39ms)
    revoke
        ✓ should fail to revoke if not admin
        ✓ should fail to claim if the id was revoked
    pause
        ✓ should fail to pause if not admin
        ✓ should fail to unpause if not admin
        ✓ should fail to claim if paused
        ✓ should be able to claim sand after pause/unpause (62ms)
    coverage
        ✓ a valid signature must verify correctly
        ✓ 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 (104ms)
      ✓ cbrt3, cube root multiplied by 1e3 (54ms)

  ERC20RewardPool main contract tests
    roles
        ✓ admin should be able to call setContributionRules (1272ms)
        ✓ other should fail to call setContributionRules (41ms)
        ✓ admin should be able to call setRewardToken
        ✓ other should fail to call setRewardToken
        ✓ admin should be able to call setStakeToken
        ✓ other should fail to call setStakeToken
        ✓ admin should be able to call setRewardCalculator
        ✓ other should fail to call setRewardCalculator
        ✓ admin should be able to call pause & unpause
        ✓ other should fail to call pause & unpause
        ✓ admin cant renounce ownership
        ✓ recoverFunds should fail if contract is not paused
        ✓ admin should be able to call recoverFunds if contract is paused (61ms)
        ✓ other should fail to call recoverFunds
        ✓ recoverFunds must fail with address zero
        ✓ contract should have enough funds to replace the stakeToken (74ms)
        ✓ contract should have enough funds to replace the rewardToken (43ms)
    reward distribution
        only one user
          ✓ reward before stake (230ms)
          ✓ stake before rewards (231ms)
          ✓ stake->withdraw so total contribution == 0, stake again (205ms)
          ✓ stake->withdraw so total contribution == 0, stake again with some rewards (263ms)
        two users
          ✓ reward before stake (233ms)
          ✓ user1 stake before rewards (280ms)
          ✓ user2 stake before rewards (274ms)
        10 users
          ✓ reward before stake (898ms)
    contribution calculation
        ✓ initial (231ms)
        ✓ computeContribution after the user change his contribution (479ms)
        ✓ computeContributionInBatch after the user change his contribution (436ms)
    contribution calculation with rewards
        ✓ initial (655ms)
        ✓ computeContribution after users change his contribution (1001ms)
        ✓ computeContributionInBatch after the user change his contribution (627ms)
    trusted forwarder and meta-tx
        ✓ should fail to set the trusted forwarder if not admin
        ✓ should success to set the trusted forwarder if a valid contract and admin
        ✓ setReward with meta-tx (53ms)
        ✓ stake with meta-tx (85ms)
        ✓ withdraw with meta-tx (110ms)

  ContributionRules
    roles
        ✓ admin should be able to call setERC721MultiplierList (1041ms)
        ✓ setERC721MultiplierList above the limits should fail (244ms)
        ✓ admin should be able to call setERC1155MultiplierList
        ✓ setERC1155MultiplierList above the limits should fail (434ms)
        ✓ admin should be able to call deleteERC721MultiplierList
        ✓ admin should be able to call deleteERC1155MultiplierList
        ✓ admin should be able to call setERC721MultiplierLimit
        ✓ admin should be able to call setERC1155MultiplierLimit
    compute contribution
        ✓ 0 ERC721 - 0 ERC1155
        ✓ 1 ERC721 balanceOf - 0 ERC1155 (120ms)
        ✓ 4 ERC721 balanceOf - 0 ERC1155 (143ms)
        ✓ 1 ERC721 balanceOf - 1 ERC1155 (151ms)
        ✓ 1 ERC721 balanceOf - 2 ERC1155 ids (165ms)
        ✓ 1 ERC721 id - 2 ERC1155 ids (86ms)
        ✓ 2 ERC721 ids - 2 ERC1155 ids (100ms)
    multiplier limit
        ✓ should limit ERC721 multiplier at 15% (232ms)
        ✓ should limit ERC1155 multiplier at 15% (59ms)
        ✓ should return MaxGlobalMultiplier (86ms)
        ✓ should not be able to set setERC721MultiplierLimit > 1000
        ✓ should not be able to set setERC1155MultiplierLimit > 1000

  ERC20RewardPool Lock Rules
      ✓ admin should be able to call setTimelockClaim (1494ms)
      ✓ other should fail to call setTimelockClaim
      ✓ should fail to setTimelockClaim above the limit
      ✓ user can only get his rewards after lockTimeMS (195ms)
      ✓ we can disable lockTimeMS check by setting it to zero (115ms)
      ✓ admin should be able to call setTimelockDeposit
      ✓ other should fail to call setTimelockDeposit
      ✓ should fail to setTimelockDeposit above the limit
      ✓ user should wait to deposit(stake) again (65ms)
      ✓ admin should be able to call setTimeLockWithdraw
      ✓ other should fail to call setTimeLockWithdraw
      ✓ should fail to setTimeLockWithdraw above the limit
      ✓ user should wait to withdraw again and exit (123ms)
      ✓ should fail to setAmountLockClaim above the limit
      ✓ should be able to claim only amount allowed or if check is disabled (113ms)

  Requirementsules
    roles
        ✓ admin should be able to call setMaxStakeOverall
        ✓ admin should be able to call setERC721RequirementList
        ✓ admin should be able to call setERC1155RequirementList
        ✓ admin should be able to call deleteERC721RequirementList (39ms)
        ✓ admin should be able to call deleteERC1155RequirementList (38ms)
    Max/Min Stake
        ✓ getERC721MaxStake should return correct values - balanceOf (46ms)
        ✓ getERC721MaxStake should return correct values - id (48ms)
        ✓ getERC721MaxStake should return correct values - id & balanceOf (59ms)
        ✓ checkERC721MinStake should fail - balanceOf (71ms)
        ✓ checkERC721MaxStake should return correct value (76ms)
        ✓ checkERC1155MaxStake should return correct value (76ms)
    Max Stake Calculator
        ✓ ERC721 balanceOf and ERC1155 (73ms)
        ✓ ERC721 balanceOf and No ERC1155 (74ms)
        ✓ ERC721 balanceId and ERC1155 (84ms)
        ✓ No ERC721 and ERC1155 (77ms)
        ✓ maxStakeOverall should cap maxStake (78ms)
    Stake
        ✓ user should be able to stake (98ms)
        ✓ stake should fail - ERC721 balanceId  (69ms)
        ✓ stake should fail - ERC721 balanceOf  (62ms)
        ✓ stake should fail - ERC1155 balanceId  (85ms)
        ✓ stake should fail - maxAllowed  (74ms)

  LandContributionCalculator
    roles
        ✓ admin should be able to call setNFTMultiplierToken (306ms)
        ✓ others should fail to call setNFTMultiplierToken (102ms)
    calculation
        ✓ zero lands
        ✓ 1 lands (66ms)
        ✓ 2 lands (79ms)
        ✓ 3 lands (92ms)
        ✓ 4 lands (95ms)
        ✓ 5 lands, to high to be minted (98ms)
        ✓ 10 lands, to high to be minted (94ms)
        ✓ 20 lands, to high to be minted (96ms)
        ✓ 50 lands, to high to be minted (94ms)
        ✓ 100 lands, to high to be minted (96ms)
        ✓ 408 lands, to high to be minted (96ms)
        ✓ 166464 lands, to high to be minted (96ms)
        ✓ 67917312 lands, to high to be minted (95ms)

  LandOwnerContributionCalculator
    roles
```

```
     ✓ admin should be able to call setNFTMultiplierToken (487ms)
     ✓ others should fail to call setNFTMultiplierToken (100ms)
  calculation
     ✓ users without lands get zero contributions
     ✓ 1 lands
     ✓ 2 lands
     ✓ 3 lands
     ✓ 4 lands (39ms)

PeriodicRewardCalculator
   ✓ only Admin can call setDuration (122ms)
   ✓ calling setDuration should update campaign duration
   ✓ 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
  reward distribution
     ✓ setup: we use the rate, so REWARDS must be multiple of duration (or leftover + reward if we add in the middle)
     ✓ we distribute rewards linearly (474ms)
     ✓ if restart is called (with contribution!=0) then rewards starts from zero again (371ms)
     ✓ calling notifyRewardAmount in the middle of the distribution will distribute the remaining + what was added (487ms)
     ✓ calling notifyRewardAmount after the distribution will distribute both amounts (508ms)

TwoPeriodsRewardCalculator
   ✓ startup (138ms)
  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
     ✓ 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 (71ms)
  reward distribution
     ✓ run an initial campaign alone (73ms)
     ✓ run initial and next campaign (136ms)
     ✓ run next campaign after the initial one finished (212ms)
  restart reward
     ✓ before everything
     ✓ in middle of the first campaign
     ✓ in middle of the second campaign
     ✓ after everything (38ms)
     ✓ intermixed (64ms)

SandRewardPool
   ✓ last time reward application should match the duration (1489ms)
   ✓ total supply is at first empty
   ✓ staking should update the reward balance, supply and staking token balance (48ms)
   ✓ withdraw should update the reward balance, supply and staking token (76ms)
   ✓ reward per token should be 0 if total supply is 0
   ✓ reward per token calculation (53ms)
   ✓ earned calculation (71ms)
   ✓ get reward should transfer the reward and emit an event (106ms)
   ✓ exiting should withdraw and transfer the reward (90ms)
   ✓ pool contains reward tokens
   ✓ user can earn reward tokens if pool has been notified of reward (59ms)
   ✓ admin can notify to start a new reward process (without sending more reward tokens)
   ✓ user cannot earn rewardTokens if they stake after the end time (38ms)
   ✓ user earns full reward amount if there is only one staker after 1 day(s) (58ms)
   ✓ user earns full reward amount if there is only one staker after 27 day(s) (54ms)
   ✓ User with 0 LAND earns correct reward amount (64ms)
   ✓ User with 0 LAND earns correct reward amount - smaller stake (73ms)
   ✓ User with 1 LAND(s) earns correct reward amount (128ms)
   ✓ User with 3 LAND(s) earns correct reward amount (180ms)
   ✓ User with 10 LAND(s) earns correct reward amount (307ms)
   ✓ User can withdraw some stakeTokens after several amounts have been staked (89ms)
   ✓ First user can claim their reward - no NFTs (85ms)
   ✓ First user can claim their reward - has NFTs (412ms)
   ✓ A user can claim their reward after multiple stakes (540ms)
   ✓ First user can exit the pool (87ms)
   ✓ A user can exit the pool after multiple stakes (138ms)
   ✓ A user with NFTs can exit the pool after multiple stakes (556ms)
   ✓ Change externals contracts
   ✓ user earnings for 0 NFT(s) match expected reward with 1 stake(s) (56ms)
   ✓ user earnings for 0 NFT(s) match expected reward with 2 stake(s) (90ms)
   ✓ user earnings for 0 NFT(s) match expected reward with 4 stake(s) (135ms)
   ✓ user earnings for 0 NFT(s) match expected reward with 10 stake(s) (294ms)
   ✓ user earnings for 1 NFT(s) match expected reward with 1 stake(s) (122ms)
   ✓ user earnings for 1 NFT(s) match expected reward with 10 stake(s) (726ms)
   ✓ user earnings for 2 NFT(s) match expected reward with 1 stake(s) (617ms)
   ✓ user earnings for 3 NFT(s) match expected reward with 1 stake(s) (1302ms)
   ✓ user earnings for 3 NFT(s) match expected reward with 10 stake(s) (922ms)
   ✓ user earnings for 89 NFT(s) match expected reward with 1 stake(s) (1734ms)
   ✓ user earnings for 89 NFT(s) match expected reward with 10 stake(s) (2507ms)
   ✓ Multiple Users' earnings for 0 NFTs match expected reward: 2 users, 10 stake each (573ms)
   ✓ Multiple Users' earnings for 0 NFTs match expected reward: 3 users, 1 stake each (134ms)
   ✓ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 1 stake each (202ms)
   ✓ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 10 stake each (210ms)
   ✓ Multiple Users' earnings for 3 NFTs match expected reward: 2 users, 1 stake each (304ms)
   ✓ Multiple Users' earnings for 100 NFTs match expected reward: 2 users, 1 stake each (3846ms)
   ✓ Staking with STAKE_AMOUNT plus an extra amount equivalent to 2 NFTs (71ms)
   ✓ Earlier staker gets more rewards with same NFT amount - small NFT number (210ms)
   ✓ Earlier staker gets more rewards with same NFT amount - large NFT number (3879ms)
   ✓ More lands give more rewards than earlier staker when NFT amounts are smaller (248ms)
   ✓ More lands do not give more rewards than earlier staker with large NFT amounts (3884ms)
   ✓ rewardToken in pool is more than amount notified (57ms)
   ✓ rewardToken in pool is zero (68ms)
   ✓ rewardToken in pool is less than amount notified (74ms)
   ✓ the call to notifyRewardAmount is made after users first call stake (63ms)
   ✓ user is earning rewards and pool is notified for a second time before end of current reward period (80ms)
   ✓ Multiplier & reward are correct (312ms)
  - 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 (1742ms)
   ✓ users without land should revert (44ms)
   ✓ if a user sells his land we can recompute the contribution (273ms)

new SandRewardPool main contract tests
  roles
     ✓ admin should be able to call setContributionCalculator (515ms)
     ✓ other should fail to call setContributionCalculator
     ✓ admin should be able to call setRewardToken
     ✓ other should fail to call setRewardToken
     ✓ admin should be able to call setStakeToken
     ✓ other should fail to call setStakeToken
     ✓ admin should be able to call setRewardCalculator
     ✓ other should fail to call setRewardCalculator
     ✓ admin should be able to call recoverFunds (46ms)
```

```
      ✓ other should fail to call recoverFunds
      ✓ recoverFunds must fail with address zero
  reward distribution
    only one user
        ✓ reward before stake (195ms)
        ✓ stake before rewards (193ms)
        ✓ stake->withdraw so total contribution == 0, stake again (154ms)
        ✓ stake->withdraw so total contribution == 0, stake again with some rewards (217ms)
    two users
        ✓ reward before stake (190ms)
        ✓ user1 stake before rewards (215ms)
        ✓ user2 stake before rewards (213ms)
    10 users
        ✓ reward before stake (762ms)
  contribution calculation
      ✓ initial (205ms)
      ✓ computeContribution after the user change his contribution (427ms)
      ✓ computeContributionInBatch after the user change his contribution (367ms)
  contribution calculation with rewards
      ✓ initial (555ms)
      ✓ computeContribution after users change his contribution (869ms)
      ✓ computeContributionInBatch after the user change his contribution (560ms)
  trusted forwarder and meta-tx
      ✓ should fail to set the trusted forwarder if not admin
      ✓ should success to set the trusted forwarder if admin
      ✓ setReward with meta-tx (50ms)
      ✓ stake with meta-tx (71ms)
      ✓ withdraw with meta-tx (85ms)

new SandRewardPool anti compound tests
    ✓ user can only get his rewards after lockTimeMS (167ms)
    ✓ we can disable lockTimeMS check by setting it to zero (95ms)
  roles
      ✓ admin should be able to call setAntiCompoundLockPeriod
      ✓ other should fail to call setAntiCompoundLockPeriod

ERC677Token
    ✓ Transfering tokens to ERC677Receiver contract should emit an OnTokenTransferEvent event (2936ms)
    ✓ Transfering tokens to EOA
    ✓ Transfering tokens to a non receiver contract should fail
    ✓ Transfering tokens to a contract with fallback function should succeed

use withSnapshot to keep your testing environment clean
    ✓ withSnapshot doesn't care about what happen before (591ms)

Faucet
    ✓ Send cannot exceed Faucet limit amout (622ms)
    ✓ Send cannot be executed twice without awaiting (128ms)
    ✓ Send succeded for correct asked amount (81ms)
    ✓ Retrieve succeded for deployer
    ✓ Retrieve succeded for deployer with any address
    ✓ Retrieve fail for user that is not deployer
    ✓ setPeriod succeed for deployer
    ✓ setPeriod fail for user that is not deployer
    ✓ setLimit succeed for deployer
    ✓ Send with new limit succeed after limit update. (88ms)
    ✓ setLimit fail for user that is not deployer

GameMinter
  GameMinter: Calling Directly
      ✓ should fail to create GAME if user has insufficient SAND (3265ms)
      ✓ should allow anyone to create a game (176ms)
      ✓ should allow owner to add assets (247ms)
      ✓ should charge a fee when owner adds assets (229ms)
      ✓ should allow editor to add assets (239ms)
      ✓ should allow owner to remove assets (287ms)
      ✓ should allow editor to remove assets (222ms)
      ✓ should fail if not authorized to add assets (183ms)
      ✓ should fail to modify GAME if user has insufficient SAND (200ms)
      ✓ should fail if not authorized to remove assets (190ms)
      ✓ should fail if not authorized to set GAME URI (180ms)
      ✓ allows GAME owner to set GAME URI (218ms)
      ✓ allows GAME editor to set GAME URI (218ms)
  GameMinter: Sandbox MetaTXs
      ✓ should allow anyone to create a game via MetaTx (348ms)
      ✓ should allow GAME Owner to add assets via MetaTx (321ms)
      ✓ should allow GAME Owner to remove assets via MetaTx (580ms)
      ✓ should allow GAME Owner to set URI via MetaTx (1767ms)
      ✓ should allow GAME Editor to add assets via MetaTx (345ms)
      ✓ should allow GAME Editor to remove assets via MetaTx (358ms)
      ✓ should allow GAME Editor to set URI via MetaTx (369ms)

GameToken
  GameToken: Minting GAMEs
      ✓ can update the GameMinter address (3063ms)
      ✓ Minter can create GAMEs when _Minter is set (134ms)
      ✓ should revert if trying to reuse a baseId (39ms)
      ✓ gameId contains creator, randomId, chainIndex & version data (72ms)
      ✓ can get the storageId for a GAME (75ms)
      ✓ can get the chainIndex for a GAME (74ms)
      ✓ reverts if non-minter trys to mint Game when _Minter is set (114ms)
    GameToken: Mint With Assets
        ✓ fails to create if "to" address is the gameToken contract (113ms)
        ✓ fails to add ERC1155 tokens to the game if Operator != GAME contract (48ms)
        ✓ fails to add ERC1155 token batch to the game if Operator != GAME contract (95ms)
        ✓ can mint Games with single Asset (152ms)
        ✓ can mint Games with many Assets (224ms)
        ✓ should fail if length of assetIds and values dont match (45ms)
    GameToken: Modifying GAMEs
        ✓ should allow the owner to add game editors (79ms)
        ✓ should allow the owner to remove game editors (80ms)
        ✓ should revert if non-owner trys to set Game Editors (59ms)
        ✓ Minter can add single Asset (212ms)
        ✓ should bump the version number in the gameId (55ms)
        ✓ Minter can add multiple Assets (215ms)
        ✓ Minter can remove single Asset (142ms)
        ✓ fails when removing more assets than the game contains (69ms)
        ✓ Minter can remove multiple Assets (212ms)
        ✓ Game token should acurately track token balances for owners (87ms)
  GameToken: Transferring GAMEs
      ✓ current owner can transfer ownership of a GAME (96ms)
      ✓ can transfer creatorship of a GAME (91ms)
      ✓ transfer creatorship should revert for a non existing game
      ✓ can transfer creatorship of a GAME back to original creator (102ms)
      ✓ should fail if non-owner trys to transfer a GAME (86ms)
      ✓ transfer creatorship should revert for a burned game (89ms)
  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 (211ms)
      ✓ fails if sender != game owner and not metatx (109ms)
    GameToken: burnAndRecover
        ✓ fails if "to" == address(0) (112ms)
        ✓ fails to destroy if "to" == Game Token contract (105ms)
        ✓ fails if "from" != game owner (107ms)
        ✓ fails if sender != game owner and not metatx (108ms)
        ✓ can destroy GAME and recover assets in 1 tx if not too many assets (193ms)
        ✓ creatorOf() should should still return original creator (105ms)
        ✓ game should no longer exist (142ms)
    GameToken: Destroy... then Recover
        ✓ fails to recover if the GAME token has not been burnt (183ms)
        ✓ can destroy without transfer of assets (226ms)
```

```
          ✓ fails to recover if "to" address is the gameToken contract (169ms)
          ✓ fails to recover assets if caller is not from or validMetaTx (177ms)
          ✓ can recover remaining assets from burnt GAME in batches (331ms)
    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
        ✓ should use baseId (creator address + subId) to map to game Assets
    GameToken: MetaTransactions
        ✓ can get isTrustedForwarder (503ms)
        ✓ can call setGameEditor via metaTx (430ms)
        ✓ can call burnFrom via metaTx (444ms)
        ✓ can call recoverAssets via metaTx (635ms)
        ✓ can call transferCreatorship via metaTx (432ms)

  AuthValidator
      ✓ signature should be valid (726ms)
      ✓ signature should be invalid

  EstateSaleWithAuth
      ✓ should be able to purchase with valid signature (5102ms)
      ✓ should NOT be able to purchase with invalid signature
      ✓ should be able to purchase through sand contract (60ms)

  LandV2
    LandBaseTokenV2
        ✓ should NOT be able to transfer 1x1 quad twice from 3x3 quad (2116ms)
        ✓ should NOT be able to transfer 1x1 quad twice from 6x6 quad (51ms)
        ✓ should NOT be able to transfer 3x3 quad twice from 6x6 quad (65ms)
        ✓ should NOT be able to transfer 1x1 quad twice from 12x12 quad (125ms)
        ✓ should NOT be able to transfer 3x3 quad twice from 12x12 quad (135ms)
        ✓ should NOT be able to transfer 6x6 quad twice from 12x12 quad (161ms)
        ✓ should NOT be able to transfer 1x1 quad twice from 24x24 quad (399ms)
        ✓ should NOT be able to transfer 3x3 quad twice from 24x24 quad (399ms)
        ✓ should NOT be able to transfer 6x6 quad twice from 24x24 quad (433ms)
        ✓ should NOT be able to transfer 12x12 quad twice from 24x24 quad (518ms)
        ✓ should return true for 1x1 quad minited inside a 3x3 quad
        ✓ should return true for 1x1 quad minited inside a 6x6 quad (42ms)
        ✓ should return true for 3x3 quad minited inside a 6x6 quad (42ms)
        ✓ should return true for 1x1 quad minited inside a 12x12 quad (111ms)
        ✓ should return true for 3x3 quad minited inside a 12x12 quad (111ms)
        ✓ should return true for 6x6 quad minited inside a 12x12 quad (112ms)
        ✓ should return true for 1x1 quad minited inside a 24x24 quad (373ms)
        ✓ should return true for 3x3 quad minited inside a 24x24 quad (1875ms)
        ✓ should return true for 6x6 quad minited inside a 24x24 quad (717ms)
        ✓ should return true for 12x12 quad minited inside a 24x24 quad (373ms)
        ✓ should return false for 1x1 quad not minited
        ✓ should return false for 3x3 quad not minited
        ✓ should return false for 6x6 quad not minited
        ✓ should return false for 12x12 quad not minited
        ✓ should return false for 24x24 quad not minited
        ✓ should revert for invalid coordinates for size 3
        ✓ should revert for invalid coordinates for size 6
        ✓ should revert for invalid coordinates for size 12
        ✓ should revert for invalid coordinates for size 24
        ✓ should revert for invalid size
        ✓ should NOT be able to transfer burned 1x1 quad twice from 3x3 quad
        ✓ should NOT be able to transfer burned 1x1 quad twice from 6x6 quad (49ms)
        ✓ should NOT be able to transfer burned 3x3 quad twice from 6x6 quad (105ms)
        ✓ should NOT be able to transfer burned 1x1 quad twice from 12x12 quad (120ms)
        ✓ should NOT be able to transfer burned 3x3 quad twice from 12x12 quad (180ms)
        ✓ should NOT be able to transfer burned 6x6 quad twice from 12x12 quad (384ms)
        ✓ should NOT be able to transfer burned 1x1 quad twice from 24x24 quad (390ms)
        ✓ should NOT be able to transfer burned 3x3 quad twice from 24x24 quad (438ms)
        ✓ should NOT be able to transfer burned 6x6 quad twice from 24x24 quad (635ms)
        ✓ should NOT be able to transfer burned 12x12 quad twice from 24x24 quad (1452ms)
        ✓ Burnt land cannot be minted again
        ✓ should not be a minter by default
        ✓ should be an admin to set minter
        ✓ should enable a minter
        ✓ should disable a minter
        ✓ should not accept address 0 as minter
        ✓ should only be able to disable an enabled minter
        ✓ should only be able to enable a disabled minter
    MetaTransactionReceiverV2
        ✓ should not be a meta transaction processor
        ✓ should enable a meta transaction processor
        ✓ should disable a meta transaction processor
        ✓ should only be a contract as meta transaction processor
        ✓ should only be the admin able to set a meta transaction processor
    AdminV2
        ✓ should get the current admin
        ✓ should change the admin to a new address
        ✓ should only be changed to a new admin
    SuperOperatorsV2
        ✓ should not be a super operator by default
        ✓ should be an admin to set super operator
        ✓ should enable a super operator
        ✓ should disable a super operator
        ✓ should not accept address 0 as super operator
        ✓ should only be able to disable an enabled super operator
        ✓ should only be able to enable a disabled super operator
    UpgradeV2
        ✓ should upgrade to V2 and keep storage intact (2474ms)
    UpgradeV3
        ✓ should upgrade to V3 and keep storage intact (2956ms)

  GAS:Multi_Giveaway_1:Claiming
      ✓ 1 claim (5688ms)
      ✓ 10 claims (733ms)
      ✓ 4000 claims (5329ms)
      ✓ 10000 claims (11720ms)
{
  "Gas per claim - 1 claim total": 223213,
  "Gas per claim - 10 claims total": 226998,
  "Gas per claim - 4000 claims total": 237071,
  "Gas per claim - 10000 claims total": 239573
}

  MerkleTree_multi
      ✓ should validate the data

  Multi_Giveaway
    Multi_Giveaway_common_functionality
        ✓ Admin has the correct role (694ms)
        ✓ Admin can add a new giveaway
        ✓ Cannot add a new giveaway if not admin (50ms)
        ✓ User can get their claimed status (521ms)
        ✓ Claimed status is correctly updated after allocated tokens are claimed - 2 claims of 2 claimed (1347ms)
        ✓ Claimed status is correctly updated after allocated tokens are claimed - 1 claim of 2 claimed (181ms)
        ✓ MultiGiveaway contract returns ERC721 received (448ms)
        ✓ 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 (47ms)
        ✓ User cannot claim sand when contract does not hold any (1020ms)
        ✓ User can claim allocated multiple tokens from Giveaway contract (1086ms)
Number of assets: 64 ; Gas used: 2392547
        ✓ User can claim allocated 64 tokens from Giveaway contract (3592ms)
        ✓ Claimed Event is emitted for successful claim (1121ms)
        ✓ User can claim allocated ERC20 from Giveaway contract when there are no assets or lands allocated (41ms)
        ✓ User cannot claim if they claim the wrong amount of ERC20
        ✓ User cannot claim more than once (124ms)
        ✓ 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 (914ms)
        ✓ 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 (51ms)
        ✓ User cannot claim from Giveaway if ERC1155 values array length does not match ids array length
        ✓ User cannot claim after the expiryTime (901ms)
        ✓ User cannot claim if expiryTime is 0
    Multi_Giveaway_two_giveaways
        ✓ User cannot claim when test contract holds no tokens - multiple giveaways, 1 claim (620ms)
        ✓ User cannot claim sand when contract does not hold any - multiple giveaways, 1 claim (1281ms)
        ✓ User can claim allocated multiple tokens from Giveaway contract - multiple giveaways, 1 claim (1324ms)
        ✓ User can claim allocated multiple tokens from Giveaway contract - multiple giveaways, 2 claims (400ms)
        ✓ 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 (232ms)
    Multi_Giveaway_single_claim
        ✓ User cannot claim when test contract holds no tokens (665ms)
        ✓ User cannot claim sand when contract does not hold any (1016ms)
        ✓ User can claim allocated multiple tokens from Giveaway contract (1045ms)
        ✓ Claimed Event is emitted for successful claim (99ms)
        ✓ User cannot claim more than once (118ms)
    Trusted_forwarder_and_meta-tx
        ✓ should fail to set the trusted forwarder if not admin (511ms)
        ✓ should succeed in setting the trusted forwarder if admin
        ✓ claim with meta-tx: user can claim from single giveaway using single claim function (1046ms)
        ✓ claim with meta-tx: user cannot claim from single giveaway using single claim function more than once (168ms)
        ✓ claim with meta-tx: user can claim from single giveaway using multiple claim function (212ms)
        ✓ claim with meta-tx: user cannot claim from single giveaway using multiple claim function more than once (169ms)
        ✓ claim with meta-tx: user can claim from multiple giveaways (1745ms)
        ✓ claim with meta-tx: user cannot claim from multiple giveaways more than once (281ms)

  Permit
      ✓ ERC20 Approval event is emitted when msg signer == owner (606ms)
      ✓ Nonce is incremented for each Approval
      ✓ 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 (43ms)
      ✓ Approved operators cannot transfer more ERC20 than their allowance
      ✓ Approved operators cannot transfer more ERC20 than there is

  Asset_ERC1155_Tunnels
    Asset <> PolygonAssetERC1155: Transfer
        ✓ cannot send asset directly to tunnel l2 (4257ms)
        ✓ cannot send asset directly to tunnel l1 (58ms)
        ✓ can transfer L2 minted asset: L2 to L1 (95ms)
        ✓ can transfer L2 minted asset of value 1: L2 to L1 (103ms)
        ✓ can transfer L2 minted assetIds from same packId: L2 to L1 (272ms)
        ✓ can transfer L2 minted assetIds from same packId more than once: L2 to L1 (342ms)
        ✓ cannot transfer more than L2 minted assets: L2 to L1 (53ms)
        ✓ can batch transfer L2 minted asset: L2 to L1 (254ms)
        ✓ transfer assets from L1 to L2 (308ms)
        ✓ transfer assets from L1 to L2 more than once (521ms)
        ✓ can transfer partial supplies of L1 minted assets: L1 to L2 (869ms)
        ✓ can transfer multiple L2 minted assets: L2 to L1 (633ms)
        ✓ can transfer partial supplies of L2 minted assets: L2 to L1 (829ms)
        ✓ can transfer assets from multiple L1 minted batches: L1 to L2 (2460ms)
        ✓ can transfer assets from multiple L2 minted batches: L2 to L1 (581ms)
        ✓ can return L1 minted assets: L1 to L2 to L1 (189ms)
        ✓ can return L2 minted assets: L2 to L1 to L2 (173ms)

  PolygonAssetERC1155.sol
    PolygonAsset: general
        ✓ user sending asset to itself keep the same balance (4628ms)
        ✓ user batch sending asset to itself keep the same balance (38ms)
        ✓ user batch sending in series whose total is more than its balance (40ms)
        ✓ user batch sending more asset than it owns should fails
        ✓ NFT index is 0 for a new ERC1155 tokenId
        ✓ can get chainIndex from tokenId
        ✓ can get the URI for an asset with amount 1 (58ms)
        ✓ can get the URI for a FT (58ms)
        ✓ fails get the URI for an invalid tokeId
        ✓ can burn ERC1155 asset (39ms)
        ✓ can mint and burn asset of amount 1 (51ms)
        ✓ can mint repeatedly (56ms)
    PolygonAsset: MetaTransactions
        ✓ can transfer by metaTx (68ms)
META TX CALL Error: OPERATOR_!AUTH
        ✓ fails to transfer someone else token by metaTx (61ms)
        ✓ can batch-transfer by metaTx (107ms)
    PolygonAsset: extractERC721From and collection information
        ✓ cannot extract ERC721 for ERC1155 supply == 1
        ✓ can extract ERC721 if ERC1155 supply > 1 (101ms)
        ✓ can extract to own address if sender == _msgSender() and supply > 1 (94ms)
        ✓ can extract to other address if sender == _msgSender() and supply > 1 (96ms)
        ✓ cannot extract to destination address if sender == _msgSender() but sender is not owner of ERC1155 (71ms)
        ✓ cannot extract to destination address if isApprovedForAll(sender, _msgSender()) but sender is not bouncer (48ms)
        ✓ can extract to destination address if isApprovedForAll(sender, _msgSender()) (104ms)
        ✓ can extract to other destination address if isApprovedForAll(sender, _msgSender()) (116ms)
        ✓ cannot extract to destination address if isApprovedForAll(sender, _msgSender()) but sender is not owner of ERC1155 (83ms)
        ✓ cannot extract ERC721 if supply == 1 if sender == _msgSender() (48ms)
        ✓ cannot extract ERC721 if supply == 1 if msgSender() is not approved operator
        ✓ can retrieve Extraction event with ERC1155 id and new ERC721 id and they are not the same as each other (84ms)
        ✓ cannot extract ERC721 if to == zeroAddress
        ✓ can extract more than once (166ms)
        ✓ can get the new ERC721 ID returned from extraction event (86ms)
        ✓ can get the new ERC721 ID returned from extraction tx (72ms)
        ✓ can check collectionOf for new ERC721 gives the ERC1155 ID (116ms)
        ✓ can still check collectionOf for new ERC721 if I burn my ERC1155 (124ms)
        ✓ can extract my last ERC1155 to an ERC721 (as long as supply was > 1 originally) (134ms)
        ✓ cannot burn ERC1155 after extraction if there is no more supply (151ms)
        ✓ can check collectionOf for new ERC721 correctly increments by 1 compared to the ERC1155 it was extracted from (113ms)
        ✓ can burn then extract and then burn some more (114ms)
        ✓ can mint multiple and extract from multiple IDs in a pack as long as the id has supply > 1 (341ms)
        ✓ cannot extract from tokenId minted with mintMultiple if supply == 1 (345ms)
        ✓ can mintMultiple and check collectionOf for a new ERC721 correctly increments by 1 compared to the ERC1155 it was extracted from (265ms)
        ✓ can see the index of my token after burning another token in the pack (169ms)
        ✓ can see my ERC721 collection information in PolygonAssetERC1155 contract even after I burn that ERC721 (167ms)
        ✓ can see that my ERC1155 tokenId was minted even after I burn all ERC1155 in that pack
        ✓ can see that my ERC1155 tokenId was minted even after I burn and extract all ERC1155 in that pack (94ms)
        ✓ can get the URI for an asset of amount 2 (124ms)
        ✓ can correctly obtain ERC721 metadata after extraction (130ms)
        ✓ get the same URI when extract a 721 (206ms)
        ✓ get the same URIs for extracted 721s from same ERC1155 collection (297ms)
        ✓ can get the chainId from extracted 721s and it does not cut across NFT_INDEX (235ms)
        ✓ can use chainIndex getter to obtain chainIndex for a given ID
        ✓ collectionOf for ERC1155 with supply == 1 is NOT equal to tokenId because IS_NFT is 1 (43ms)
        ✓ collectionOf for ERC1155 with supply > 1 is equal to tokenId

  Asset_ERC721_Tunnels
    AssetERC721 <> PolygonAssetERC721: Transfer
      L1 to L2
          ✓ if not owner cannot pause tunnels (3147ms)
          ✓ if not owner cannot unpause tunnels
          ✓ owner can set Max Limit on L1
          ✓ cannot set Max Limit on L1 if not owner
          ✓ should not be able to transfer AssetERC721 when paused (111ms)
          ✓ cannot tranfer asset directly to tunnel l1
          ✓ cannot tranfer asset directly to tunnel l2
          ✓ should be able to transfer multiple assets to L2 (1358ms)
          ✓ should be able to transfer multiple assets to L2 - higher start id (1352ms)
      L2 to L1
          ✓ if not owner cannot pause tunnels
          ✓ only owner can pause tunnels
          ✓ if not owner cannot unpause tunnels
          ✓ only owner can unpause tunnels
          ✓ should not be able to transfer AssetERC721 when paused (196ms)
          ✓ should be able to transfer 1 AssetERC721 (101ms)
          ✓ should should be able to transfer multiple assets (940ms)
          ✓ should not be able to transfer if exceeds limit (274ms)

    PolygonAssetERC721.sol differences with AssetERC721.sol
      roles
        admin
          ✓ admin role is set (430ms)
        MINTER
          ✓ check initial roles
          ✓ minter can mint tokens
          ✓ other user should fail to mint

    PolygonBundleSandSale.sol
      ✓ should fail to deploy with a zero receiving wallet (144ms)
      ✓ 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.
      ✓ 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 ? (53ms)
        ✓ NFT can be used with numPacks == 1  (61ms)
        ✓ single sale (66ms)
        ✓ multiple/batch sale (136ms)
      Withdraw
        ✓ withdraw (280ms)
        ✓ should fail to withdraw if not admin (135ms)
      Buy packs with DAI
        ✓ should fail with the wrong saleId
        ✓ should fail if not enough packs (165ms)
        ✓ should fail if not enough DAI (183ms)
        ✓ buy with DAI, obs: buyer != to (307ms)
        Buy packs with ETH
          ✓ should fail with the wrong saleId
          ✓ should fail if not enough packs (167ms)
          ✓ should fail if not enough ETH (185ms)
          ✓ buy with ETH, obs: buyer != to (298ms)

    AssetAttributesRegistry
      ✓ getRecord for non existing assetId (10046ms)
      ✓ setCatalyst for legendary catalyst with 4 gems (162ms)
      ✓ setCatalyst should fail for non minter account
      ✓ setCatalyst with gems.length > MAX_NUM_GEMS should fail (232ms)
      ✓ setCatalyst with gems.length > maxGemForCatalyst should fail (156ms)
      ✓ setCatalystWithBlockNumber should fail for non migration contract
      ✓ addGems to rareCatalystId (173ms)
      ✓ should fail for non-nft
      ✓ addGems should fail for non minter account (119ms)
      ✓ addGems should fail for empty gemsId array (70ms)
      ✓ addGems should fail for non existing catalystId (68ms)
      ✓ should fail for gemId = 0
      ✓ addGems should fail when trying to add two gems in total to commonCatalyst (151ms)
      ✓ admin can change attributes contract
      ✓ fails if anyone other than admin trys to change attributes (46ms)

    AssetAttributesRegistry: getAttributes
      getAttributes: minting
        ✓ can get attributes for 1 gem (13249ms)
        ✓ can get attributes for 2 identical gems (469ms)
        ✓ can get attributes for 3 identical gems (487ms)
        ✓ can get attributes for 4 identical gems (511ms)
        ✓ can get attributes for 2 different gems (474ms)
        ✓ can get attributes for 3 different gems (490ms)
        ✓ can get attributes for 4 different gems (519ms)
        ✓ can get attributes for 2 identical gems + 1 different gem (495ms)
        ✓ can get attributes for 3 identical gems + 1 different gem (521ms)
        ✓ can get attributes for 2 identical gems + 2 different identical gems (507ms)
      getAttributes: upgrading
        ✓ can get attributes when adding 1 gem to an asset with an empty catalyst (511ms)
        ✓ can get attributes when adding 2 identical gems to an asset with an empty catalyst (2327ms)
        ✓ can get attributes when adding 3 identical gems to an asset with an empty catalyst (528ms)
        ✓ can get attributes when adding 4 identical gems to an asset with an empty catalyst (546ms)
        ✓ can get attributes when adding 2 different gems to an asset with an empty catalyst (509ms)
        ✓ can get attributes when adding 3 different gems to an asset with an empty catalyst (532ms)
        ✓ can get attributes when adding 4 different gems to an asset with an empty catalyst (549ms)
        ✓ can get attributes when adding 1 similar gem to an asset with existing gems (529ms)
        ✓ can get attributes when adding 1 different gem to an asset with existing gems (527ms)
        ✓ can get attributes when adding 2 similar gems to an asset with existing gems (550ms)
        ✓ can get attributes when adding 2 different gems to an asset with existing gems (547ms)
        ✓ can get attributes when adding 3 similar gems to an asset with existing gems (559ms)
        ✓ can get attributes when adding 3 different gems to an asset with existing gems (569ms)
        ✓ can get attributes when adding gems to an asset multiple times (601ms)
        ✓ can get attributes when upgrading an asset multiple times (1393ms)
        ✓ attributes after multiple upgrades are correct (608ms)
        ✓ should fail if numGems > MAX-NUM_GEMS (511ms)

    AssetMinter
      AssetMinter: Mint
        ✓ the assetMinterAdmin is set correctly (13448ms)
        ✓ the assetMinter quantities are set correctly (87ms)
        ✓ mintWithCatalyst transaction reverts in case of change in quantitiesByCatalyst (13134ms)
        ✓ mintWithCatalyst transaction reverts in case of change in numberOfGemsBurnPerAsset
        ✓ mintWithCatalyst transaction reverts in case of change in numberOfCatalystBurnPerAsset
        ✓ Record is created with correct data on minting with legendary catalyst (NFT) (103ms)
        ✓ Transfer event is emitted on minting an NFT (catalyst legendary) - TODO: fix (1260ms)
        ✓ CatalystApplied event is emitted on minting an NFT with a catalyst (1430ms)
        ✓ Catalysts and gems totalSuplies are reduced when added (1500ms)
        ✓ Mint without catalyst reverts in case of change in quantitiesByAssetType (1255ms)
        ✓ Mint without catalyst (105ms)
        ✓ Mint custom number admin (128ms)
        ✓ Mint custom number user3 (154ms)
        AssetMinter: MintMultiple
          ✓ TransferBatch event is emitted on minting a single FT via mintMultiple (1405ms)
          ✓ TransferBatch event is emitted on minting a single FT via mintMultipleWithoutCatalyst (71ms)
          ✓ TransferBatch event is emitted on minting a multiple FTs (615ms)
          ✓ TransferBatch event is emitted on minting a multiple FTs via mintMultipleWithoutCatalyst (137ms)
          ✓ CatalystApplied event is emitted for each NFT minted with a catalyst (1972ms)
          ✓ records should be updated correctly for each asset minted (461ms)
          ✓ totalSupply & balance should be reduced for burnt gems & catalysts (1384ms)
        AssetMinter: addGems
          ✓ Can extract an erc721 & add Gems (15298ms)
        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 (274ms)
          ✓ should fail if gemIds.length > maxGems (130ms)
          ✓ 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
          ✓ mintMultipleWithoutCatalyst should fail if supplies.length == 0
          ✓ mintMultipleWithoutCatalyst should fail if mintData.to is zero address
          ✓ mintMultipleWithoutCatalyst should fail if mintData.from is not message sender
          ✓ mintMultipleWithoutCatalyst should fail if supplies array has wrong value
          ✓ mintMultipleWithoutCatalyst should fail if supplies array and assetTypeIds array has difrent length
          ✓ mintMultiple should fail if catalystsQuantities == 0 (116ms)
          ✓ 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 (1337ms)
          ✓ mintMultiple: should fail if assets length and supplies length differ (1310ms)
          ✓ mintMultiple: should fail if numberOfCatalystBurnPerAsset differ (108ms)
          ✓ mintMultiple: should fail if numberOfGemsBurnPerAsset differ (108ms)
          ✓ mintMultiple: should fail if supplies has in valid values (99ms)

    AssetUpgrader
      ✓ extractAndSetCatalyst for FT with rareCatalyst and powerGem, no ownership change (9057ms)
      ✓ setting a rareCatalyst with powerGem and defenseGem (258ms)
      ✓ adding powerGem and defenseGem to a rareCatalyst with no gems (302ms)
      ✓ setting a rareCatalyst where ownerOf(assetId)!= msg.sender should fail (190ms)
      ✓ burns sand fees when feeRecipient = BURN_ADDRESS (250ms)

    GemsCatalystsRegistry
      ✓ getMaxGems for commonCatalyst should be 1 (9872ms)
      ✓ can get decimals
      ✓ getMaxGems for non existing catalystId should fail
      ✓ burnCatalyst should burn 2 common catalysts from catalystOwner account (91ms)
      ✓ Allow max value allowance for every gems and catalyst (107ms)
      ✓ Allow 0 allowance for every gems and catalyst (154ms)
      ✓ burnCatalyst should fail for unauthorized account
      ✓ burnCatalyst should fail for non existing catalystId
      ✓ burnCatalyst should fail for insufficient amount (78ms)
      ✓ burnCatalyst should fail for account with no gems (70ms)
      ✓ burnGem should burn 3 power gems from gemOwner account (84ms)
      ✓ burnGem should fail for unauthorized account
      ✓ burnGem should fail for non existing gemId
      ✓ burnGem should fail for insufficient amount (71ms)
      ✓ burnGem should fail for account with no gems (69ms)
      ✓ addGemsAndCatalysts should fail for existing gemId
      ✓ addGemsAndCatalysts should fail for existing catalystd
      ✓ addGemsAndCatalysts should fail for catalyst with address zero
      ✓ addGemsAndCatalysts should fail for gem with address zero
      ✓ addGemsAndCatalysts should add gemExample
      ✓ addGemsAndCatalysts should add catalystExample
      ✓ addGemsAndCatalysts should fail for gem id not in order
      ✓ addGemsAndCatalysts should fail for unauthorized user (49ms)
      ✓ addGemsAndCatalysts should fail if too many G&C (357ms)
      ✓ addGemsAndCatalysts pass if max -1 G&C (355ms)
      ✓ batchBurnGems reverts on gemsIds and amounts length mismatch (70ms)
      ✓ batchBurnGems for two different gem tokens (120ms)
      ✓ batchBurnCatalysts reverts on catalystIds and amounts length mismatch (74ms)
      ✓ batchBurnCatalysts for two different catalyst tokens (112ms)
      ✓ batchBurnGems for two different gem tokens and two different amounts (105ms)
      ✓ only DEFAULT_ADMIN_ROLE can set trusted forwarder (58ms)
      ✓ cannot set trustedForwarder to zero address

    MockLandWithMint.sol
      ✓ creation (769ms)
      ✓ cannot set polygon Land Tunnel to zero address (4252ms)
      ✓ supported interfaces
      Mint and transfer full quad
        With approval
          ✓ transfers quads of all sizes (8866ms)
        Without approval
          ✓ reverts transfers of quads (3173ms)
        From self
          ✓ should NOT be able to transfer burned quad twice through parent quad (16811ms)
          ✓ should NOT be able to transfer burned 1x1 through parent quad (283ms)
          ✓ transfers of quads of all sizes from self (5159ms)
      Burn and transfer full quad
        ✓ should revert transfer quad from zero address
        ✓ should revert transfer quad to zero address
        ✓ burnt token cannot be approved (46ms)
        With approval
          ✓ should not transfer a burned 1x1 quad (54ms)
          ✓ should not transfer burned quads (15890ms)
        From self
          ✓ should not transfer a burned 1x1 quad (65ms)
          ✓ should not transfer burned quads (14259ms)
      mint and check URIs
        ✓ mint and check URI 1 (53ms)
        ✓ mint and check URI 3 (63ms)
        ✓ mint and check URI 6 (162ms)
        ✓ mint and check URI 12 (569ms)
        ✓ mint and check URI 24 (2083ms)
        ✓ reverts check URI for non existing token
      Mint and transfer a smaller quad
        ✓ transferring a 1X1 quad from a 3x3 (80ms)
        ✓ transferring a 1X1 quad from a 12x12 (550ms)
        ✓ transferring a 3X3 quad from a 6x6 (204ms)
        ✓ transferring a 6X6 quad from a 12x12 (2393ms)
      Mint and transfer all its smaller quads
        ✓ transferring all 1X1 quad from a 3x3 (343ms)
        ✓ transferring all 1X1 quad from a 6x6 (1168ms)
        ✓ transferring all 1X1 quad from a 12x12 (4801ms)
      transfer batch
        ✓ transfers batch of quads of different sizes (5229ms)
        ✓ transfers batch of quads of different sizes from self (7165ms)
        ✓ 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 (38ms)
        ✓ Transfer 1x1 with approval (48ms)
      testing batchTransferFrom
        ✓ Mint 12x12 and transfer all internals 1x1s from it (1398ms)
      Meta transactions
        transferQuad without approval
  META TX CALL Error: not authorized to transferQuad
  META TX CALL Error: not authorized to transferQuad
  META TX CALL Error: not authorized to transferQuad
  META TX CALL Error: not authorized to transferQuad
  META TX CALL Error: not authorized to transferQuad
          ✓ should not transfer quads of any size (10025ms)
        transferQuad with approval
          ✓ should transfer quads of any size (16270ms)
        transferQuad from self
          ✓ should revert transfer of quad twice through parent quad (24129ms)
          ✓ should transfer quads of any size (10789ms)
        Burn and transfer full quad
          ✓ should revert transfer of 1x1 quad after burn (146ms)
          ✓ should revert transfer of quad if a sub quad is burned (40703ms)
          ✓ should revert transfer of any size quad after burn (56389ms)
        batchTransferQuad
          ✓ should batch transfer 1x1 quads (214ms)
          ✓ should batch transfer quads of different sizes (2374ms)
      Getters
        ✓ returns the width of the grid
        ✓ returns the height of the grid
        ✓ should fetch x and y values of given quad id (8427ms)
        ✓ cannot fetch x and y values of given non existing quad id
        ✓ should fetch owner of given quad id (6469ms)
        ✓ checks if a quad is valid & exists (2839ms)

    PolygonLand
      Land <> PolygonLand: Transfer
        L1 to L2
          ✓ only owner can pause tunnels (4862ms)
          ✓ cannot accept randomly transferred land (51ms)
          ✓ cannot accept randomly transferred lands as batch
          ✓ only owner can unpause tunnels
          ✓ set Max Limit on L1
          ✓ cannot set Max Limit on L1 if not owner
          ✓ set Max Allowed Quads
          ✓ cannot Max Allowed Quads if not owner
          ✓ cannot set Max Allowed Quads to zero
          ✓ should not be able to transfer Land when paused (140ms)
          ✓ should be able to transfer 1x1 Land (116ms)
          ✓ should be able to transfer 3x3 Land (173ms)
          ✓ should be able to transfer 6x6 Land (876ms)
          ✓ should be able to transfer 12x12 Land (2886ms)
```

```
         ✓ should be able to transfer 24x24 Land (4420ms)
         ✓ should should be able to transfer multiple lands (511ms)
        Through meta transaction
           ✓ should be able to transfer 1x1 Land (140ms)
           ✓ should be able to transfer 3x3 Land (194ms)
           ✓ should be able to transfer 6x6 Land (399ms)
           ✓ should be able to transfer 12x12 Land (1218ms)
           ✓ should should be able to transfer multiple lands meta (532ms)
      L2 to L1
         ✓ only owner can pause tunnels
         ✓ cannot accept randomly transferred land (115ms)
         ✓ cannot accept randomly transferred lands as batch (124ms)
         ✓ only owner can unpause tunnels
DUMMY CHECKPOINT. moving on...
         ✓ should not be able to transfer Land when paused (211ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 1x1 Land (188ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 12x12 Land (2125ms)
         ✓ should not be able to transfer 2, 12x12 Land at once (4442ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 3x3 Land (293ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 6x6 Land (663ms)
DUMMY CHECKPOINT. moving on...
         ✓ should should be able to transfer multiple lands (872ms)
         ✓ should not be able to transfer if exceeds limit (199ms)
      Through meta Tx
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 1x1 Land (210ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 3x3 Land (311ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 6x6 Land (680ms)
DUMMY CHECKPOINT. moving on...
         ✓ should be able to transfer 12x12 Land (2141ms)

  PolygonLand:WithAdminV2
     ✓ should get the current admin
     ✓ should change the admin to a new address
    Meta transactions
       ✓ should change the admin to a new address (45ms)

  PolygonLand:WithSuperOperatorsV2
     ✓ should not be a super operator by default
     ✓ should be an admin to set super operator
     ✓ should enable a super operator
     ✓ should disable a super operator
    Meta transactions
       ✓ should enable a super operator (47ms)

  PolygonLandWeightedSANDRewardPool
     ✓ last time reward application should match the duration (3656ms)
     ✓ total supply is at first empty
     ✓ staking should update the reward balance, supply and staking token balance (49ms)
     ✓ withdraw should update the reward balance, supply and staking token (68ms)
     ✓ reward per token should be 0 if total supply is 0
     ✓ reward per token calculation (43ms)
     ✓ earned calculation (64ms)
     ✓ get reward should transfer the reward and emit an event (93ms)
     ✓ exiting should withdraw and transfer the reward (83ms)
     ✓ pool contains reward tokens
     ✓ user can earn reward tokens if pool has been notified of reward (55ms)
     ✓ admin can notify to start a new reward process (without sending more reward tokens)
     ✓ user cannot earn rewardTokens if they stake after the end time (38ms)
     ✓ user earns full reward amount if there is only one staker after 1 day(s) (58ms)
     ✓ user earns full reward amount if there is only one staker after 27 day(s) (49ms)
     ✓ User with 0 LAND earns correct reward amount (56ms)
     ✓ User with 0 LAND earns correct reward amount - smaller stake (56ms)
     ✓ User with 1 LAND(s) earns correct reward amount (138ms)
     ✓ User with 3 LAND(s) earns correct reward amount (208ms)
     ✓ User with 10 LAND(s) earns correct reward amount (357ms)
     ✓ User can withdraw some stakeTokens after several amounts have been staked (77ms)
     ✓ First user can claim their reward - no NFTs (73ms)
     ✓ First user can claim their reward - has NFTs (357ms)
     ✓ A user can claim their reward after multiple stakes (541ms)
     ✓ First user can exit the pool (86ms)
     ✓ A user can exit the pool after multiple stakes (133ms)
     ✓ A user with NFTs can exit the pool after multiple stakes (616ms)
     ✓ Change externals contracts
     ✓ user earnings for 0 NFT(s) match expected reward with 1 stake(s) (58ms)
     ✓ user earnings for 0 NFT(s) match expected reward with 2 stake(s) (71ms)
     ✓ user earnings for 0 NFT(s) match expected reward with 4 stake(s) (122ms)
     ✓ user earnings for 0 NFT(s) match expected reward with 10 stake(s) (278ms)
     ✓ user earnings for 1 NFT(s) match expected reward with 1 stake(s) (121ms)
     ✓ user earnings for 1 NFT(s) match expected reward with 10 stake(s) (795ms)
     ✓ user earnings for 2 NFT(s) match expected reward with 1 stake(s) (156ms)
     ✓ user earnings for 3 NFT(s) match expected reward with 1 stake(s) (180ms)
     ✓ user earnings for 3 NFT(s) match expected reward with 10 stake(s) (981ms)
     ✓ user earnings for 89 NFT(s) match expected reward with 1 stake(s) (2003ms)
     ✓ user earnings for 89 NFT(s) match expected reward with 10 stake(s) (2874ms)
     ✓ Multiple Users' earnings for 0 NFTs match expected reward: 2 users, 10 stake each (533ms)
     ✓ Multiple Users' earnings for 0 NFTs match expected reward: 3 users, 1 stake each (112ms)
     ✓ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 1 stake each (222ms)
     ✓ Multiple Users' earnings for 1 NFTs match expected reward: 2 users, 10 stake each (222ms)
     ✓ Multiple Users' earnings for 3 NFTs match expected reward: 2 users, 1 stake each (341ms)
     ✓ Multiple Users' earnings for 100 NFTs match expected reward: 2 users, 1 stake each (4389ms)
     ✓ Staking with STAKE_AMOUNT plus an extra amount equivalent to 2 NFTs (65ms)
     ✓ Earlier staker gets more rewards with same NFT amount - small NFT number (226ms)
     ✓ Earlier staker gets more rewards with same NFT amount - large NFT number (4637ms)
     ✓ More lands give more rewards than earlier staker when NFT amounts are smaller (279ms)
     ✓ More lands do not give more rewards than earlier staker with large NFT amounts (4899ms)
     ✓ rewardToken in pool is more than amount notified (51ms)
     ✓ rewardToken in pool is zero (65ms)
     ✓ rewardToken in pool is less than amount notified (70ms)
     ✓ the call to notifyRewardAmount is made after users first call stake (65ms)
     ✓ user is earning rewards and pool is notified for a second time before end of current reward period (72ms)
     ✓ Multiplier & reward are correct (360ms)
     ✓ Only sender or reward distribution can compute sender's account

  PolygonSANDRewardPool
     ✓ last time reward application should match 30 days (1497ms)
     ✓ total supply is at first empty
     ✓ staking should update the reward balance, supply and staking token balance (41ms)
     ✓ withdraw should update the reward balance, supply and staking token (58ms)
     ✓ reward per token should be 0 if total supply is 0
     ✓ reward per token calculation (40ms)
     ✓ earned calculation (56ms)
     ✓ get reward should transfer the reward and emit an event (66ms)
     ✓ exiting should withdraw and transfer the reward (55ms)

  RaffleCareBears
     ✓ should be able to mint with valid signature (1734ms)
    - should be able to mint 20_000 different tokens
    - should be able to mint 20_000 different tokens in 3 waves
    - should be able to mint 20_000 different tokens in 3 waves in 3 txs
     ✓ should be able to personalize with valid signature (146ms)
     ✓ should not be able to personalize with invalid signature (125ms)
     ✓ should be able to differentiate a personalized asset (2697ms)
     ✓ should not be able to personalize twice with the same signature (306ms)

  PolygonSand.sol Meta TX
    transfer
       ✓ without metatx
       ✓ with metatx (60ms)
    approve and transferFrom
       ✓ without metatx (38ms)
       ✓ with metatx (99ms)
    burn
       ✓ without metatx
       ✓ with metatx (48ms)
    trusted forwarder
       ✓ should fail to set the trusted forwarder if not owner
       ✓ should success to set the trusted forwarder if owner
    approveAndCall
       ✓ without metatx
       ✓ with metatx (48ms)
    paidCall
       ✓ without metatx
       ✓ with metatx (56ms)

  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 (5812ms)
       ✓ should be able to transfer SAND: L2 to L1 (5993ms)
    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 (594ms)
    Transfers
       ✓ users should be able to transfer some of the token they have
       ✓ users should be able to transfer all token they have
       ✓ 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
    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 (1156ms)
     ✓ reverts if claim amount is more than balance
     ✓ returns the amount of sand which has been claimed

  SandPolygonDepositor
     ✓ Locking funds in sand predicate mock contract (1160ms)

  PolygonStarterPack.sol
    PurchaseValidator.sol
       ✓ can get the backend signing wallet (10792ms)
       ✓ default admin can set the backend signing wallet
       ✓ a SigningWallet event is emitted when the signing wallet is updated
       ✓ cannot set the signing wallet to zeroAddress
       ✓ if not default admin cannot set the signing wallet (53ms)
    Roles
       ✓ default admin should be set
       ✓ starterpack admin should be set
    Setup
       ✓ correct receiving wallet has been implemented
       ✓ check the domain separator
       ✓ check the chainId
    getReceivingWallet
       ✓ can view the receiving wallet
    setReceivingWallet
       ✓ default admin can set the receiving wallet
       ✓ a ReceivingWallet event is emitted when the receiving wallet is updated
       ✓ cannot set the receiving wallet to zeroAddress
       ✓ if not default admin cannot set the receiving wallet (54ms)
    setSANDEnabled
       ✓ STARTERPACK_ROLE can set SAND enabled
       ✓ DEFAULT_ADMIN_ROLE can set SAND enabled/disabled (but only because sandAdmin is currently the same as starterPackAdmin)
       ✓ SandEnabled event is emitted when SAND is enabled
       ✓ SandEnabled event is emitted when SAND is disabled
       ✓ if not STARTERPACK_ROLE cannot set SAND enabled (52ms)
       ✓ STARTERPACK_ROLE can disable SAND
       ✓ if not STARTERPACK_ROLE cannot disable SAND (53ms)
    setPrices
       ✓ STARTERPACK_ROLE can set the prices for all cats and gems (58ms)
       ✓ DEFAULT_ADMIN_ROLE can set the prices (but only because sandAdmin is currently the same as starterPackAdmin) (66ms)
       ✓ an individual catalyst price can be updated
       ✓ an individual gem price can be updated
       ✓ if not STARTERPACK_ROLE cannot set prices (45ms)
       ✓ cannot set prices for cat that does not exist
       ✓ cannot set prices for gem that does not exist
       ✓ cannot set prices for cat 0
       ✓ cannot set prices for gem id 0
       ✓ SetPrices event is emitted when prices are updated (74ms)
       ✓ SetPrices event is emitted when a single price is updated
       ✓ default admin cannot set prices within delay period (73ms)
       ✓ default admin can set the prices again after the delay period (136ms)
    withdrawAll
       ✓ default admin can withdraw remaining cats and gems from contract (183ms)
       ✓ default admin cannot withdraw remaining cats and gems from contract to zeroAddress
       ✓ default admin receives correct cats and gems balances upon withdrawal (214ms)
       ✓ if not default admin cannot withdraw any cats and gems from contract (52ms)
       ✓ cannot withdraw cats that do not exist
       ✓ cannot withdraw gems that do not exist (120ms)
       ✓ withdrawal does not fail for zero balances if id exists (437ms)
       ✓ withdrawAll event is emitted upon withdrawal (176ms)
    purchaseWithSAND
       ✓ can purchase bundle of cats and gems when SAND is enabled - zero prices (186ms)
       ✓ can purchase bundle of cats and gems when SAND is enabled and prices are >0 with correct SAND amount (327ms)
       ✓ cannot purchase bundle of cats and gems when SAND is not enabled
[
  '0xa0Ee7A142d267C1f36714E4a8F75612F20a79720',
  [ 1, 2, 3, 4 ],
  [
    BigNumber { _hex: '0x01', _isBigNumber: true },
    BigNumber { _hex: '0x01', _isBigNumber: true },
    BigNumber { _hex: '0x01', _isBigNumber: true },
    BigNumber { _hex: '0x01', _isBigNumber: true }
  ],
  [ 1, 2, 3, 4, 5 ],
  [
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true }
  ],
  BigNumber { _hex: '0x00', _isBigNumber: true },
  buyer: '0xa0Ee7A142d267C1f36714E4a8F75612F20a79720',
  catalystIds: [ 1, 2, 3, 4 ],
  catalystQuantities: [
    BigNumber { _hex: '0x01', _isBigNumber: true },
    BigNumber { _hex: '0x01', _isBigNumber: true },
    BigNumber { _hex: '0x01', _isBigNumber: true },
    BigNumber { _hex: '0x01', _isBigNumber: true }
  ],
  gemIds: [ 1, 2, 3, 4, 5 ],
  gemQuantities: [
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true },
    BigNumber { _hex: '0x02', _isBigNumber: true }
  ],
  nonce: BigNumber { _hex: '0x00', _isBigNumber: true }
] message
       ✓ a successful purchase results in a Purchase event (270ms)
       ✓ cannot purchase bundle of cats and gems without enough SAND (133ms)
       ✓ cannot purchase bundle of cats and gems if have not approved the StarterPack contract (120ms)
       ✓ cannot purchase bundle of cats and gems if StarterPack contract does not have any (301ms)
       ✓ purchase fails with incorrect backend signature (44ms)
       ✓ purchase fails with bad message params - catalyst lengths
       ✓ purchase fails with bad message params - gem lengths
       ✓ purchase invalidates the nonce after 1 use (276ms)
       ✓ cannot purchase cats that do not exist (176ms)
       ✓ cannot purchase gems that do not exist (193ms)
       ✓ cannot purchase if not msgSender() (87ms)
       ✓ purchase occurs with old prices if price change has not yet taken effect (286ms)
       ✓ purchase occurs with updated prices after price change delay (354ms)
       ✓ purchase occurs with first price change before second price change has taken effect (347ms)
       ✓ allows multiple nonce queues for a given buyer (690ms)
       ✓ order of cat IDs should not matter (424ms)
       ✓ order of gem IDs should not matter (2457ms)
       ✓ can get nonce for a buyer
       ✓ cannot reuse nonce (245ms)
    getPrices
       ✓ cats and gems prices are initially 0 (with 0 switchTime)
       ✓ cats and gems prices can be viewed after an update has been made (177ms)
    isSANDEnabled
       ✓ can view whether SAND is enabled or not
    metatransactions
       ✓ can purchase with metatx (261ms)
    test array lengths for withdrawAll
       ✓ can withdraw 20 types of gems (6569ms)
       ✓ can withdraw 20 types of cats and gems (13536ms)
       ✓ cannot withdraw more than the limit of catalysts and gems (21063ms)
    GAS:PolygonStarterPack-PurchaseWithSAND
{
  "Gas - WithdrawAll 100 each of 20 cats and 100 each of 20 gems - ": 2003357
}
       ✓ WithdrawAll gas used for 100 each of 20 cats and 100 each of 20 gems (13799ms)
{
  "Gas - WithdrawAll 100 each of 30 cats and 100 each of 30 gems - ": 2984210
}
       ✓ WithdrawAll gas used for 100 each of 30 cats and 100 each of 30 gems (26766ms)
{
  "Gas - WithdrawAll 100 each of 50 cats and 100 each of 50 gems - ": 4946009
}
       ✓ WithdrawAll gas used for 100 each of 50 cats and 100 each of 50 gems (46435ms)
{
  "Gas - WithdrawAll 100 each of 50 cats - ": 2491673
}
       ✓ WithdrawAll gas used for 100 each of 50 cats only (41571ms)
{
  "Gas - WithdrawAll 100 each of 50 gems - ": 2491861
}
       ✓ WithdrawAll gas used for 100 each of 50 gems only (41530ms)

  RafflePeopleOfCrypto
     ✓ should be able to mint with valid signature (1601ms)
    - should be able to mint 20_000 different tokens
    - should be able to mint 20_000 different tokens in 3 waves
    - should be able to mint 20_000 different tokens in 3 waves in 3 txs
     ✓ should be able to personalize with valid signature (141ms)
     ✓ should not be able to personalize with invalid signature (120ms)
     ✓ should be able to differentiate a personalized asset (162ms)
     ✓ should not be able to personalize twice with the same signature (130ms)

  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

  RaffleSteveAoki
     ✓ should be able to mint with valid signature (1624ms)
    - should be able to mint 20_000 different tokens
    - should be able to mint 20_000 different tokens in 3 waves
    - should be able to mint 20_000 different tokens in 3 waves in 3 txs
     ✓ should be able to personalize with valid signature (140ms)
     ✓ should not be able to personalize with invalid signature (134ms)
     ✓ should be able to differentiate a personalized asset (167ms)
     ✓ should not be able to personalize twice with the same signature (140ms)

  ERC20BasicApproveExtension
     ✓ ApproveAndCall calling buyLandWithSand (4513ms)
     ✓ ApproveAndCall should fail for input data too short (39ms)
     ✓ ApproveAndCall should fail for first parameter != sender
     ✓ ApproveAndCall should fail for zero data
     ✓ ApproveAndCall should fail for Approving the zeroAddress
     ✓ ApproveAndCall should work for target = EOA with ether value = 1
     ✓ ApproveAndCall should work for target = EOA with ether value = 0 (39ms)
     ✓ ApproveAndCall for an empty contract as a target should revert
     ✓ ApproveAndCall calling logOnCall of a mock contract (52ms)
     ✓ ApproveAndCall with only one parameter should fail
     ✓ ApproveAndCall calling revertOnCall of a mock contract should fail
     ✓ PaidCall calling buyLandWithSand (404ms)
     ✓ PaidCall should fail for input data too short
     ✓ PaidCall should fail for first parameter != sender
     ✓ PaidCall should fail for zero data
     ✓ PaidCall should fail for Approving the zeroAddress
     ✓ PaidCall should work for target = EOA with ether value = 1
     ✓ PaidCall should work for target = EOA with ether value = 0 (44ms)
     ✓ PaidCall for an empty contract as a target should revert
     ✓ PaidCall calling logOnCall of a mock contract (58ms)
     ✓ PaidCall calling revertOnCall of a mock contract should fail

  Gems & Catalysts permit
     ✓ user can use permit function to approve Gems via signature
     ✓ user can use permit function to approve Catalysts via signature
     ✓ 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 (61ms)
    Transfers
       ✓ users should be able to transfer some of the token they have
       ✓ users should be able to transfer all token they have
       ✓ 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
    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 (620ms)
     ✓ nonAtomicBatchWithETH
     ✓ atomicBatch
     ✓ nonAtomicBatch
     ✓ singleTargetAtomicBatchWithETH
     ✓ singleTargetNonAtomicBatchWithETH
     ✓ singleTargetAtomicBatch
     ✓ singleTargetNonAtomicBatch
```



```
    ✓ onERC1155Received
    ✓ onERC1155BatchReceived
    ✓ onERC721Received
    ✓ supportsInterface


1665 passing (18m)
18 pending
```

Code Coverage

Initial audit
Quantstamp usually recommends developers increase the branch coverage to `90%` and above before a project goes live to avoid hidden functional bugs that might not be easy to spot during the development phase. For branch code coverage, the current targeted files by the audit achieve a lower score that needs to be improved before deployment.
Fix review: After reviewing the fixes, the branch coverage scores are still low and need to be improved.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
solc_0.6/EstateSale/					
AuthValidator.sol	88.89	50	75	88.89	20
EstateSaleWithAuth.sol	79.03	44.74	62.5	79.03	… 118,176,208
solc_0.6/Interfaces/					
LandToken.sol	100	100	100	100	
solc_0.6/ReferralValidator/					
ReferralValidator.sol	14	6.25	20	14	… 187,201,206
solc_0.6/common/BaseWithStorage/					
Admin.sol	0	0	0	0	… 25,26,30,31
MetaTransactionReceiver.sol	40	0	33.33	40	19,20,32
solc_0.6/common/Interfaces/					
ERC1155.sol	100	100	100	100	
ERC20.sol	100	100	100	100	
solc_0.6/common/Libraries/					
SigUtil.sol	0	0	0	0	… 48,49,51,57

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

```
3d7641223b6fb227e52f9f675f2efb96e5ad8dfda2e958b5e7a0b6fe68c9f431 ./solc_0.6/ReferralValidator/ReferralValidator.sol
295f5dc69ed8d2200f897fd45e46cfc2a27f790d277c1caa556833ee5e4e1ea1 ./solc_0.6/EstateSale/EstateSaleWithAuth.sol
8ae036053b1745fb4f89359ce966af19673030e708fb585b5bcbf14f7097b387 ./solc_0.6/EstateSale/LandToken.sol
1621d2d6f982d74f3750213f1587b30c92b88c4675a29ce80e1b7dcb8b190570 ./solc_0.6/EstateSale/AuthValidator.sol
166876143ed91353301a821db176d54a3d1f4757ec911d55b6b8e8ecb4e5a254 ./solc_0.6/Interfaces/LandToken.sol
29ceb8bda3bdabbaadd6198a0523472f17ecfb9f5f5cec3696648ea0f0974269 ./solc_0.6/common/BaseWithStorage/Admin.sol
2c2c2f3e3247938a3feda4375cf2729fc6f19d724dbc69a4499fc1758232e9ef ./solc_0.6/common/BaseWithStorage/MetaTransactionReceiver.sol
7e67f61a70549319921432d7064228d44c3b13ac4a60d69472bf9f110dd89712 ./solc_0.6/common/Interfaces/ERC20.sol
fd441460320f43831bc2e06c838808bf0cb3cb7b27f47e0403ac7804ff14cfcb ./solc_0.6/common/Interfaces/ERC1155.sol
258c59f5dd22a6966aa46f6d07fcd7895d335e792f91b7232956bb84a56b37dd ./solc_0.6/common/Libraries/SigUtil.sol
```

Tests

```
e92ce1e5512bb7257ca1f6e916137d4a8017c106476261ed099f91790cb8337c ./test/evmSnapshotRevert.test.ts
202266a6189fcff67bcc7ce9e59a463de535fa4d4a6798e6a1b16b75ca2c6ad8 ./test/chaiBigNumberCloseTo.ts
c1b5c8c0197083d7216bab7d946557171356e5546f90b26209eec31240c888be ./test/forwardRequestData712.ts
0add2aa5be824aad14efda2b0a1654649e2d4fe24eec4d2d6419f4215ec7b965 ./test/utils.ts
90c62410cb80c9281b2e4f1429973f7f23ac2f674676e3f7f51bed6e068d68f8 ./test/sendMetaTx.ts
4e2585a6ddb2147ba235c09d16feee3b1e16264ccbfbf04b7ebf9a17b248c841 ./test/chai-setup.ts
c383c55aa73898de1da2dff6e9299f7c5efe3c61dbfabda457b2099a88685d80 ./test/erc721/index.js
```

```
cb019cdadcb0720c95436db9dc00799227d1a201e7c26980b7117cc79abc1b7d ./test/permit/fixtures.ts

27e78a5c3fb2b0df5fb1c7f165985b2701a2ba09902408c77aa19863e58e5bca ./test/permit/permit.test.ts

e0648ed03953d4cd58e92be5b0c4efe5e45def6e8167b45e73bbef5c884d8c7d ./test/permit/data712.ts

1206bd54f0adad846c9506f1464c0a37a31482a11c247d5e3193f24534ff141e ./test/permit/data712Upgradeable.ts

ee022b3c676541fc4662d00f93c7b29d842c5cd73035301aa20a1ecfa676ce7d ./test/claim/signedGiveaway/fixtures.ts

ca8447e666f25ec322cb764b4a36e580066bc242bf8485dfe2754031604d6f9e ./test/claim/signedGiveaway/signature.ts

28a1f0c47624a3564667acf9a099ade7c7bb806cb46fff4241b73fcd5df0a6a0 ./test/claim/signedGiveaway/signedGiveaway.ts

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

64c700386c99b2191d7fe3109c0b33b967ff85d0833cbb30289dd6de6e90a685 ./test/polygon/starterPack/fixtures.ts

f2e8cd97e8abf1f2d521a4c133b822407488548a637a55b3fc9a0f7bb1690e29 ./test/polygon/starterPack/signature.ts

7a84c11dd1b1eec31eede72b90c37208620f5804999c92e716d097d23bf9dd07 ./test/polygon/starterPack/polygonStarterPack.test.ts

f8cb80f0925c3d58ae8ba5de0ecab493765ba5d3b1f1929c685062c10a3408f4 ./test/polygon/land/fixtures.ts

925b8c5cd2716b146f652e20a91956df6a99238d3aa812794ef35ec8a19d9b32 ./test/polygon/land/withSuperOperatorsV2.test.ts

b74ff596751a50378bff6462a8167f255c97471ef16b0aef145724355abca480 ./test/polygon/land/withAdminV2.test.ts

bc6a67d4e466a5e78462908c2908b2ccb9241ac1f33b0853ba00c23c39cf09d0 ./test/polygon/land/testERC721.test.js

cc447f15499fda8fffde1ae1440f1eacc1b75daf5716eb3c5c2d7ac22a901d81 ./test/polygon/land/land.test.ts

fcaedae21691944551870897bdeaa3288ef889232a24633bf7ab61a235a506c8 ./test/polygon/land/landTunnels.test.ts

8fef8814cefe7e1f0ad7b5cb819ae0f356fc02b621f4f61b9239fe57635a4a90 ./test/polygon/asset/fixtures.ts

8a00ea66343aa6602e2e411fa4c3efa669f85b8085a0faa8bd15d098df654d5a ./test/polygon/asset/assetERC1155Tunnels.test.ts

925e3189539cc34ac60b248d72274b4ac19fb482ab88d4896065594e3e73c916 ./test/polygon/asset/fixtures_tunnels.ts

6ad726de0d3026cbbe9b4af0e401d049afd5259d104562b5b77180e4e952980e ./test/polygon/asset/polygonAssetERC1155.test.ts

c5607cc578228d0fe1acb8eb2b7fef5abe906344c468dff7631fd55873ff1af7 ./test/polygon/asset/fixtures_auction712Data.ts

8309dec1a662cb91bb9c7ec68fe63866c5338e39dc4733966d6ef623ef384f96 ./test/polygon/catalyst/utils.ts

bf179cbea72e501622d5bc18e83e8915c0e9624a2fc1e5229c29536223093348 ./test/polygon/catalyst/assetMinter/fixtures.ts

44071f19480d0ffd6f1374430ee259f49e4321ad14c06acab7c3bc6cf67a8513 ./test/polygon/catalyst/assetMinter/assetMinter.test.ts

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

d8803078646f3732f4220cc3758e64a11b395f9d485b1edc7ec964fe44e828e2
./test/polygon/catalyst/collectionCatalystMigrations/collectionCatalystMigrations.test.ts

80690ec592655672c2d93055247864088115c040f3c1a0b9f4db26b513b1fcdc ./test/polygon/catalyst/assetAttributesRegistry/fixtures.ts

7c8782f94f0df0c689263a4ba9ac7d4cb3f13d441c6a737ec4af2849af41096c ./test/polygon/catalyst/assetAttributesRegistry/getAttributes.test.ts

23e8e25415a9e8e665ad3730504fdfe9285e7593d1840ab3443b6c8af65b7cfe
./test/polygon/catalyst/assetAttributesRegistry/assetAttributesRegistry.test.ts

8d58a6a70035f808a5f5472b29d9e819aeb5633e24a8b734fe48128dc1381364 ./test/polygon/catalyst/assetUpgrader/assetUpgrader.test.ts

32710c5512c5649adf8158e4df7101e9f6cda5af2e847f96e8e2073eebf90680 ./test/polygon/catalyst/gemsCatalystsRegistry/gemsCatalystsRegistry.test.ts

c613099698ee90e80879079d1fe4940b424e3db52e265db3a5a21fc3dfe6c300 ./test/polygon/catalyst/gemsCatalystsRegistry/gems.test.js

c08ab254816ac5e78e4fe60d3025f89e7a8957a7a2504a53e7ec033cd6607175 ./test/polygon/catalyst/gemsCatalystsRegistry/catalyst.test.js

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

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

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

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

b076909ab47a2d30ced823b6ede3ca1a77b4805a62f5948fd711f5dfae6a4578 ./test/polygon/assetERC721/fixtures.ts

4be982f367a9ef6c8fda4f623ef50996ab9af5e8154b92c29a212eabf457712c ./test/polygon/assetERC721/polygonAssetERC721.test.ts

446621b640271c8d32953dcb6848f2b67cb8c91dae5c3868b102ba301a39dfd8 ./test/polygon/assetERC721/assetERC721Tunnels.test.ts

e569687498200a2d148caeb911f14907113fc31022acaf912d57118acaa7029b ./test/polygon/raffle/CareBears/CareBears.fixtures.ts

a40b8941bbea41e4f520fe85cf11a3f2723ca28f5adca5870372a5eb0f5bb0a6 ./test/polygon/raffle/CareBears/CareBears.test.ts

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

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

4b782e35829eca22a4eea19ec51cf004c0ba159207d03f1951eda2b98cd3931c ./test/polygon/liquidityMining/polygonSANDRewardPool.test.ts

0d8a1264dc88ced7a315b42dc9c42a2cead9a1ba616fe86fb692a75e38f451b3 ./test/polygon/bundleSandSale/fixtures.ts

6ae5a5e07966f07b9a352771f21bcb51ed6fcee5ebc5c943ab83446be09ceb2c ./test/polygon/bundleSandSale/bundleSandSale.test.ts

8d06b89cfb35aa01aab286012cf36701ba27de1f4e5171ccb702541b0c7ddc2b ./test/polygon/sand/fixtures.ts

2f60fc2fb3543b80ef4cdae994029370656132dcc19f426f8c50d70a5026a931 ./test/polygon/sand/sand-metatx.test.ts

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

3a274b58576e167a0768f57bec009ab1763a072501f43b65d0c0addd45e4d678 ./test/erc677/fixtures.ts

e5a805a4cd1ab6014a612a2f1477c2a07466b0decb56dce231fd24e6550b4f6f ./test/erc677/ERC677.test.ts

4343b8aecafb6e569485a723db137d8d7aead24a86c9ea4587b5881bf1135419 ./test/erc1155/index.js

8a09f6ac2f66832841547c7f57df3b9a0d8589ceaab1380b0bae0e91994a76b1 ./test/utils/batch.test.ts

5f3d19fc9cbc0b43c6ac9435c91497a54ac1b4dd7cafdee1e7cb2281fc75bbe4 ./test/land/fixtures.ts

e584afd3552a3e25382610bc78d2d3b1fd0f2df4974fda754ddf89401dcb9592 ./test/land/testERC721.test.js

15c32cfede35e25a49b223e3d194ad9b6e29a31cc66fd638284af14d6ae2e808 ./test/land/land.test.ts
```

```
f445ccc7e231da18f4d03dc88f5305c75b81809d7f8c06f6318ed614e88f59ec ./test/asset/fixtures.ts

f60a5bbb3be9fcf215d7a5afa717b5b3996b8f1ebe66ded15716cbb6d149d40f ./test/asset/asset.test.ts

845d1b50b1d6595b87c07bf60ea9ba971b1fa486ca145e7d88bfe1cbef703b63 ./test/asset/assetERC1155_genericERC1155.test.js

baa8901658696435895d6e64b59a98bea427bee2743d1b51011834c23f6bbfc8 ./test/asset/assetSignedAuctionWithAuth.test.ts

a6af6cafd822aabad11bff64aa220a44d1d7692ff7f6fb396ea26a930087f48d ./test/asset/originalAsset.test.ts

c1bb310b4b5883c26a76009e8ac8b5036f60a5c47cf2c693b05f7bb6f6bec659 ./test/asset/assetERC721_genericERC721.test.js

3730aaa115a7fa4e64346bc2f534adeead6466764bc3a775d6dfa080d28b8fd1 ./test/asset/assetERC721.test.ts

ae4f18e4f66e37f5ce4611cd4ba419e92f9ecbf5f656b089d0f8a7cab9a00508 ./test/asset/polygonAssetERC1155_genericERC1155.test.js

c0cf8d1ec195ea04c77717a8de2d2d8a34d1e3eb054d84f0cc50d2c429f7eabc ./test/defi/erc20RewardPool/erc20RewardPool.test.ts

8b2e3797d44f0d8b5eb71a3db4619a3db41121240c7be6ee4daf4764d1dcaa1e ./test/defi/erc20RewardPool/fixtures/fixtures.ts

4b6cc9e7b2ae910f95e4961c19961b0f9a43899a710314630267d0cff1c96d8b ./test/defi/erc20RewardPool/rules/RequirementsRules.test.ts

fcfbfc4b0374d61f3cb931851367f5c3c8ff74cba3c7fe35f0cf5e292132888b ./test/defi/erc20RewardPool/rules/LockRules.test.ts

67cf0fdc41c5b1302597a2c4cb90db54499f5259d9139e1b8eea30689df82e49 ./test/defi/erc20RewardPool/rules/ContributionRules.test.ts

c2e31c1e43465663db1f22325878f30a02da46d382118321d08ec9c24474a17e ./test/defi/sandRewardPool/utils.ts

c38b3efc3aeab6823ba606c2d4e6631baf11db822d617c072d960e40986d81f4 ./test/defi/sandRewardPool/originalSet.test.ts

082b4da1ad32bfeffffeee59d97849a0603dacb38c1dd90727ab2877c91e630c ./test/defi/sandRewardPool/sandRewardPoolAntiCompund.test.ts

2cbfb916332015acbe4397cce356e3899d3a69d6342cc4059d665eed01c9a25f ./test/defi/sandRewardPool/sandRewardPool.test.ts

f4fb9c8cc47ec789b9731ed8e8b870d53b8da01870cb7aa8bdf86133632ea2e2 ./test/defi/sandRewardPool/sandRewardPool.e2e.test.ts

f71066f11fd8a404d783d5e7513facd61df4b01a562ff7f6eb2335d37311c320 ./test/defi/sandRewardPool/fixtures/rewardCalculator.fixture.ts

ab61d477415ac2c7ccb4950bc62b885d01acece29213aae4e2f834b84e62b4ec ./test/defi/sandRewardPool/fixtures/contributionCalculator.fixture.ts

aa7353b2c49a3dcffad0a9ebdf4e4a0f61aaba07a46a9ddc0120d84427108edf ./test/defi/sandRewardPool/fixtures/originalFixtures.ts

87717154cdbe0797a6f3a88c154365cbcf845b2716b12249914cbc5cfff0212c ./test/defi/sandRewardPool/fixtures/sandRewardPool.fixture.ts

5fd3515e65eb7d29f2dfc7c7bc5053e6a69eba0cdfb843273a16203ae2c36c4a ./test/defi/sandRewardPool/calculators/periodicRewardCalculator.test.ts

19dc982906fd0afb2f5060eac9756aaf9494c9c46f88a17898038b18b908655e
./test/defi/sandRewardPool/calculators/landOwnerContributionCalculator.test.ts

dce71f8644f7a3b55a3764b895b5627b0ece6c6a494abac9e734aa68533d563d ./test/defi/sandRewardPool/calculators/landContributionCalculator.test.ts

efc6dccc492f78d0ca6a8f05eedd3147e64bfb05e68be5ed8115cb093a789cf7 ./test/defi/sandRewardPool/calculators/twoPeriodsRewardCalculator.test.ts

75515bd70629a9f2533fdd11be2ae7ec5216734419f84dbbf00b6cc1f34fe122 ./test/faucet/fixtures.ts

6d00a2d48163c5a504702f421b872bd258f4294174fd9092ac180b9140ad3d0b ./test/faucet/faucet.test.ts

83245ee8e74a849058f0679196b7ee79468e7a30da99d9e4b6908bd2b6d32c0d ./test/assetGiveaway/fixtures.ts

01ee014a7bb074dc88f10485612bc362faaf994019685f2f51de63852574452a ./test/assetGiveaway/gas.test.ts

0e631c666c0ed83c69931627790dcfb40f24f98b0b21ce183377894ebb88f818 ./test/assetGiveaway/assetGiveaway.test.ts

cbcb3b04e57d4056958eb618bc4d1c67ba7d6fe91bdb59561f210215d34e7f1c ./test/assetGiveaway/merkleTree.test.ts

b2a2b836d163d6fe29d53a48ad8f81b2f268e4c9c1d42b0964272767fdf19089 ./test/land-sale/fixtures.ts

47fb504a1e638d4f7db4bee1406ae2bbdb7357eadfed33c56f3bb0f6587753ac ./test/land-sale/EstateSaleWithAuth.test.ts

4ef0f64fe9b9c4f95f81ca4aefc80e68ed8251abe92f4a86b72c122feaa3385b ./test/land-sale/AuthValidator.test.ts

306e15dbbf40f2dcbd3442d9fb1825b92cfb5a4153c003533112d514ee934a8d ./test/erc20/index.js

3392e856aec29290ad5be72d496068d46cdf9040e0558bb7cd83eac930e49e6b ./test/raffle/PeopleOfCrypto/PeopleOfCrypto.fixtures.ts

7a79e45da7796d1bf8d87e79e3645f00d516f04f1377b868869a87a0dea131f0 ./test/raffle/PeopleOfCrypto/PeopleOfCrypto.test.ts

2a29bd9dfe3004c8dc666438bbef87a90f8132c76abfd98b343ddbfe145a8809 ./test/raffle/SnoopDogg/fixtures.ts

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

ab3e46f1c09b4fea32edfffd12a8623f6320e69202242cc121ccb58f2a05790d ./test/raffle/SteveAoki/SteveAoki.test.ts

12f1cd46f47eee120a2a0778ed01870875e6e7892ec3b4af76d603b92f04edf2 ./test/raffle/SteveAoki/SteveAoki.fixtures.ts

03d7855ef6e22c20254d7bf63f190365bb17873646734bf8c12038dc207e0e98 ./test/liquidityMining/contributionEquation.test.js

11d7ac4fb2c753e8fbadafbaa312597129dc2a2478f3ec62f18f1e9afa23733c ./test/liquidityMining/SANDRewardPool.test.js

4c1bd4e849b1d1527e8760bdd197727d53c0782f63c1ab8b4ef3106c42450f67 ./test/liquidityMining/mockSANDRewardPool.test.js

5ae7b1e3b742b624b07e2c84a9fd7dba1a1815801b1cc1a98350ff7754c3603e ./test/liquidityMining/_testHelper.js

0a471ff6ded5b2fa335fda291c792a498d5b01139c7ac3a1bd4206bd51e1598d ./test/sand/fixtures.ts

7e9cde00c035f0e9c825a880fd4e196f352f61bcd27829d7025944a855031804 ./test/sand/erc20BasicApproveExtension.test.ts

86a380fdd6fbd4d7dbc1e423cc405af1d6a8fb3487c725d3d858da121bfa93fe ./test/sand/ERC20Permit.test.ts

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

7e4337fd6647813c944303328f2aac1171ad171fb2d73e9b9229b511b4440049 ./test/multiGiveaway/fixtures.ts

d8337ff0c027f771b549306859151a60ffc86978aefe4bfa283a4c9fd3a1a26b ./test/multiGiveaway/balanceHelpers.ts

ea43bf3dcf7acd814bcc92f89cee2eec8eb9110fd954e7c6df2ee1d737ff55b5 ./test/multiGiveaway/gas.test.ts

0712516d945dc9e1bebd88e105dccfddefadbe7cbf2df92b161ffc5354262240 ./test/multiGiveaway/multiGiveaway.test.ts

40119487b7614d062cce7fab896f84c2b989c0f2939f855b48a3e21b9eb38811 ./test/multiGiveaway/merkleTree.test.ts

e1097fde2100ae26f0abd99b9794b9f4726831f112846c4e8101c488882d143c ./test/common/contributionEquation.ts

ce7090737921ca5fa6db7355539e8c3165dfefde7b1c88a17daed0da5f1ebbca ./test/common/fixtures/erc20BasicApproveExtension.ts

04b3d1f5b595c93c71e25b69d629c0eea49cb2ca68baf17617e60169fd9d607c ./test/common/fixtures/assetUpgrader.ts
```

```
385c3ebc4c3374d4c95655358b44efef52c3424ba2c514ac73fcccb5f48633f2 ./test/common/fixtures/asset.ts
04649e1c8c5e70ab63ce8fbf1b1fb1528e67a87c2674527610d8986eb08ac2b1 ./test/common/fixtures/gemAndCatalysts.ts
0bfbdceed97b800c296de01a719a6dd0af7b258e4284cac54c3ff9b29bda97b1 ./test/common/fixtures/assetAttributesRegistry.ts
7064c6690285ddadec0aef9e9abe8c1f9f53ef97f145ba3924844aed4ca13ff8 ./test/common/fixtures/assetERC721.ts
f5461b339b1849ced82083e51c9d71fd262a0de851159e5b722a4fd467a23951 ./test/common/libraries/safeMathWithRequire.ts
dab2aa28cbdae90b8a0184313d8ce3a4f4645527b451bd841d2c9f3919e5d67a ./test/Game/fixtures.ts
c1b5c8c0197083d7216bab7d946557171356e5546f90b26209eec31240c888be ./test/Game/data712.ts
3e4a967d3ae5d82e4a16a15538fcfd469fd9d077a8101d7c1ef052363d414ff2 ./test/Game/assets.ts
4420763b5282e59a6de50ed92affa850d56faec988225f9973b570a992cb6e6d ./test/Game/GameToken.test.ts
287170d7dfc9480014eafeb92bf9ba289d04ff5a9c93f18310f5cfd1e82a12fe ./test/Game/testERC721.test.js
2d3a7ced5bd7ee47474c2d2674e3c5159d7a369ef7118d3d1c01c34fb5221aed ./test/Game/gameMinter/gameMinter.test.ts
```

Changelog

- 2022-10-18 - Initial report
- 2022-11-07 - Fix review (9c7d06c)

About Quantstamp

Quantstamp is a global leader in blockchain security backed by Pantera, Softbank, and Commonwealth among other preeminent investors. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its white glove security and risk assessment services.

The team consists of web3 thought leaders hailing from top organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Many of the auditors hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 250 audits and secured over $200 billion in digital asset risk from hackers. In addition to providing an array of security services, Quantstamp facilitates the adoption of blockchain technology through strategic investments within the ecosystem and acting as a trusted advisor to help projects scale.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Aave, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT