Quantstamp Security Assessment Certificate

April 14th 2022 — Quantstamp Verified

Hike - Rush Gaming

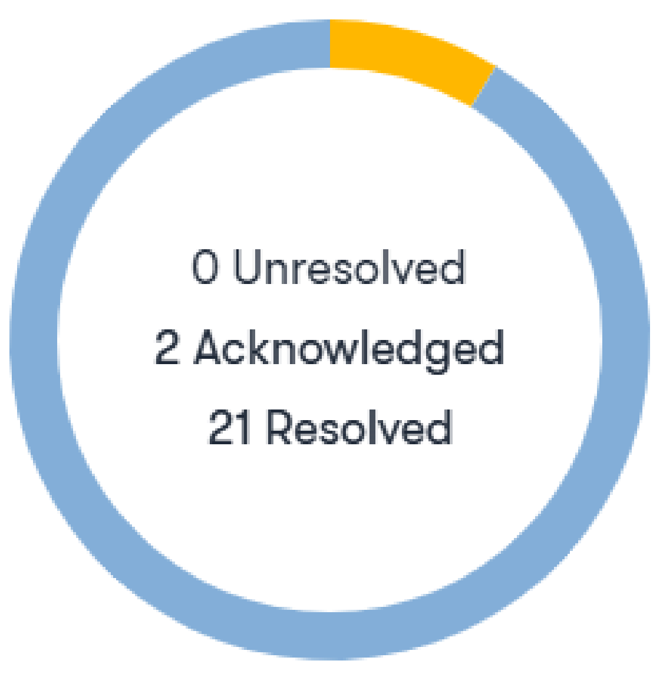
This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Wallet and NFT contract
Auditors	Souhail Mssassi, Research Engineer Philippe Dumonet, Senior Research Engineer Marius Guggenmos, Senior Research Engineer
Timeline	2022-03-07 through 2022-03-14
EVM	Berlin
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Hike Documentation
Documentation Quality	Medium
Test Quality	Medium
Source Code	

Repository	Commit
smartcontracts	ded4678
smartcontracts	61cf850

Total Issues	**23**	(21 Resolved)
High Risk Issues	**1**	(1 Resolved)
Medium Risk Issues	**3**	(3 Resolved)
Low Risk Issues	**11**	(10 Resolved)
Informational Risk Issues	**7**	(6 Resolved)
Undetermined Risk Issues	**1**	(1 Resolved)



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

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

Summary of Findings

Initial audit:
Through reviewing the code, we found **23 potential issues** of various levels of severity:1 high-severity, 3 medium-severity, 11 low-severity, 7 informational-severity and 1 undetermined issues. We recommend addressing all the issues before deploying the code.
After reaudit:
Quantstamp has checked the commit hash `61cf850` and has determined that all the reported issues have been resolved (that is, either fixed or acknowledged) by the team. More details regarding each of the issues are provided in the update messages below each issue recommendation.

ID	Description	Severity	Status
QSP-1	Owner Has Excessive Privileges Over RushToken	High	Fixed
QSP-2	Locked ETH In The Vesting Contract	Medium	Fixed
QSP-3	Any User Can Release Tokens	Informational	Fixed
QSP-4	`_totalTokensToBeUnlocked` And `_unlockedIndexes` Not Updated Upon Removing Vesting Party	Medium	Fixed
QSP-5	`ContractTokenUnlockManager`: Owner Can Remove / Add Vested Parties Or Drain Contract At Any Time	Medium	Fixed
QSP-6	Owner Can Add Beneficiary Multiple Times	Low	Fixed
QSP-7	Lack Of Events For Critical State Changes	Low	Fixed
QSP-8	Incorrect Result In The Unlocked Tokens	Low	Fixed
QSP-9	Missing Input Verification	Low	Fixed
QSP-10	Missing Address Validation	Low	Fixed
QSP-11	Incompatibility With Deflationary Tokens	Low	Fixed
QSP-12	For Loop Over Dynamic Array	Low	Fixed
QSP-13	Owner Can Renounce Ownership	Low	Fixed
QSP-14	Floating Pragma	Low	Fixed
QSP-15	Using `transfer` To Send Ether Might Revert	Low	Fixed
QSP-16	Approve Race	Informational	Acknowledged
QSP-17	`getBeneficiaries` May Run Out Of Gas	Informational	Fixed
QSP-18	Duplication Of Access Control Logic	Informational	Fixed
QSP-19	Token ID Not Human Legible In Revert String	Informational	Fixed
QSP-20	Comments Left In the Code	Informational	Fixed
QSP-21	Replace Custom Linked List Implementation With Library	Informational	Fixed
QSP-22	Multi Recipient ERC1155 Transfer Method May Not Work	Low	Acknowledged
QSP-23	`Rush1155Token`: Potential Proxy Implementation Not Initializable	? Undetermined	Fixed

Quantstamp Audit Breakdown

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

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

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

Methodology

The Quantstamp auditing process follows a routine series of steps:

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

Toolset

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

Setup

Tool Setup:

- Slither v0.8.1

Steps taken to run the tools:

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

Findings

QSP-1 Owner Has Excessive Privileges Over RushToken

Severity: *High Risk*

Status: Fixed

File(s) affected: `RushToken.sol`

Description: The `owner` address of the contract can specify a `_trustedForwarder` address which can imitate any address allowing the "trusted forwarder" to execute any transaction such as token transfers on behalf of users or even transferring ownership over the contract itself. Beyond updating the "trusted forwarder" which can imitate any address the owner address can also update the contract logic, meaning they could increase or even unlock the mint cap and even remove user's tokens. Use of an upgradeable proxy comes at the large cost of reducing trustlessness and most likely security.

Recommendation:

- Prevent the forwarder from being updated at all, have it set to an immutable value in the constructor or instantiate a fresh `MinimalForwarder` contract in the constructor and store it
- Evaluate whether upgradeability of the overall contract logic is necessary

Update: The team has fixed the issue of the trusted forwarder based on our recommendation. Only the owner / upgradeability part of the issue remains unfixed knowing that the upgradeability of the ERC20 contract is a business requirement.

QSP-2 Locked ETH In The Vesting Contract

Severity: Medium Risk

Status: Fixed

File(s) affected: `VestingWallet.sol`

Description: The contract level comment in `VestingWallet` states this contract handles the vesting of ETH and ERC20 tokens for a given beneficiary. Accordingly, it contains a payable receive function to accept ether. The `release` function, however, is only able to handle ERC20 tokens and thus any ether sent to this contract will end up locked.

Recommendation:

1. Add support for vesting and releasing ether. OR
2. Remove the payable receive function to no longer accept ether.

Update: The team has fixed the issue by removing the `receive` function.

QSP-3 Any User Can Release Tokens

Severity: Informational

Status: Fixed

File(s) affected: `VestingWallet.sol`

Description: The `release` method allows any address to call it, but the documenting `README.md` file states: "Only an admin can add an unlock config to the contract for any beneficiary and release unlocked tokens to beneficiary".

Recommendation: Ensure that only the owner can call the `release` method or clarify that any party should be able to trigger a release in the documentation.

Update: The team updated their documentation to reflect that anyone can release vested tokens.

QSP-4 `_totalTokensToBeUnlocked` And `_unlockedIndexes` Not Updated Upon Removing Vesting Party

Severity: Medium Risk

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`

Description: When a vesting account is removed from the contract by deleting its config via the `removeLockedAmountConfig` method, certain variables are not updated. Specifically, the `_unlockedIndexes` of the account is not reset and `_totalTokensToBeUnlocked` is not decreased by the amount that was not yet released by the account. This has two significant consequences:

1. If the account gets added again, it would not be able to release tokens before the previous unlocked index
2. due to `_totalTokensToBeUnlocked` not being reduced, certain tokens would permanently be stuck in the contract upon removal.

Recommendation: Have `_unlockedIndexes` of the beneficiary be reset via `delete _unlockedIndexes[beneficiary]` and reduce `_tokenTokensToBeUnlocked` by the amount the `beneficiary` has not yet claimed. This can be queried via the existing `getTokensToBeUnlocked` method.

Update: The team has removed the functionality of deleting vesting accounts.

QSP-5 `ContractTokenUnlockManager`: Owner Can Remove / Add Vested Parties Or Drain Contract At Any Time

Severity: Medium Risk

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`

Description: The owner address of the `ContractTokenUnlockManager` contract can at any time remove vested parties. Tokens that would already be subject to release are not released to the beneficiary upon removal. This may be problematic as beneficiaries have no guarantees within the contract whether they will be able to get tokens that were allocated to them. The owner could also at any time create a new vesting schedule that allows them to withdraw any remaining tokens in the contract.

Recommendation: Limit the owner addresses's ability to remove vested parties: either by removing the ability to remove vested parties altogether or by having removal require the consent of the beneficiary.

Update: The team has fixed the issue and now the beneficiaries can no longer be removed.

QSP-6 Owner Can Add Beneficiary Multiple Times

Severity: Low Risk

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`

Description: The `ContractTokenUnlockManager` keeps track of all the list of beneficiaries of the contract via a counter `_beneficiaryCount` and a linked list of beneficiaries `_beneficiaries`. However, nothing prevents the owner address from calling `addLockedAmountConfig` twice for the same beneficiary and creating a loop in the list of beneficiaries. There is a check whether the `isAdded` flag of a beneficiary has been set to `true` but it doesn't have to be set by the caller.

Recommendation: Ensure that the `isAdded` flag is set to true in the `addLockedAmountConfig` method.

Update: The team has fixed the issue by tracking the beneficiaries via `EnumerableSet.AddressSet` of OpenZeppelin and duplicate entries are no longer possible.

QSP-7 Lack Of Events For Critical State Changes

Severity: Low Risk

Status: Fixed

File(s) affected: `RushToken.sol`, `ContractSpenderManager.sol`

Description:

- `RushToken` does not emit an event when the "trusted forwarder" is changed. It is good practice to emit events on critical state changes, as it allows simpler tracking of these changes off-chain.
- `ContractSpenderManager` does not emit any events when adding or removing spenders.

Exploit Scenario: The team has fixed the issue by adding the necessary events.

Recommendation:

- `contracts/RushToken.sol`: Add an event for when the "trusted forwarder" changes and have it be emitted in the `setTrustedForwarder` method.
- `contracts/ContractSpenderManager.sol`: Add an event for when the "spender" changes.

Update: The team has resolved the issue by adding the necessary events.

QSP-8 Incorrect Result In The Unlocked Tokens

Severity: Low Risk

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`

Description: In the two parameter version of the `getTokensToBeUnlocked` method (L59-62) the `currentIndex` is set as `0`. This means that the method returns all the tokens that would be claimable at the specified `timestamp` if the user hasn't claimed anything. However, certain tokens may already be released leading to the result being inaccurate.

Recommendation: Use the `_unlockedIndexes` map to get the current index for the address being queried. Alternatively if including released tokens in the total is desired, this fact should be explicitly documented as the name of the method may be misleading.

Update: The function was renamed to `getTokensVestingSchedule` to clear up the confusion.

QSP-9 Missing Input Verification

Severity: Low Risk

Status: Fixed

File(s) affected: `VestingWallet.sol`

Description: The `_start` timestamp may be in the past upon contract deployment. This can lead to the beneficiary being able to directly access any tokens entrusted to the contract. (VestingWallet.sol [L30])

Recommendation: Consider validating the `_start` variable and comparing it with the current time using `block.timestamp`.

Update: The team has fixed the issue by verifying the `_start` variable

QSP-10 Missing Address Validation

Severity: Low Risk

Status: Fixed

File(s) affected: `CustodialWallet.sol`, `RushToken.sol`

Description: Certain functions lack a safety check in the address. The address-type argument should include a zero-address test. Otherwise, the contract's functionality may become inaccessible.

- `CustodialWallet.withdrawMoneyTo(_to)` (L44);
- `CustodialWallet.withdrawMoneyTo(_to)` (L52);

Recommendation: It's recommended to further validate certain parameters, such as addresses. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Update: The team has fixed the issue by adding the verification for address 0.

QSP-11 Incompatibility With Deflationary Tokens

Severity: Low Risk

Status: Fixed

File(s) affected: `ContractUnlockTokenManager.sol`

Description: In the `release` function `(L51)`, when transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation: Add necessary mitigation mechanisms to keep track of accurate balances. One possibility is to query the balance before and after the transfer to compute the actual delta.

Update: The team has fixed the issue by calculating the difference between `balanceOf` before and after the transfer to the account.

QSP-12 For Loop Over Dynamic Array

Severity: Low Risk

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`, `CustodialWalletFactory.sol`

Description: When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial of Service attack. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

- `ContractTokenUnlockManager.getTokensToBeUnlocked` (L68);

Recommendation: Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocks and thus multiple transactions.

Update: The team has fixed this issue by limiting the size of the array to be less than `MAXIMUM_LOCKED_AMOUNTS`.

QSP-13 Owner Can Renounce Ownership

Severity: Low Risk

Status: Fixed

File(s) affected: `ContractSpenderManager.sol`, `ContractUnlockTokenManager.sol`, `CustodialWalletFactory.sol`

Description: Several contracts implement OpenZeppelin's `Ownable`, which by default provides the function `renounceOwnership` to relinquish the ownership of the contract. In case it is never planned that the contracts should be without an owner, we recommend overwriting this function to avoid accidentally leaving the contracts without an owner.

Recommendation: Consider whether renouncing the ownership is a valid use case and disable the functionality by overwriting `renounceOwnership` in case it is not.

Update: The team has fixed the issue by overriding the `renounceOwnership` function to always make it revert.

QSP-14 Floating Pragma

Severity: Low Risk

Status: Fixed

File(s) affected: `ContractSpendable.sol`, `ContractSpenderManager.sol`, `ContractUnlockTokenManager.sol`, `CustodialWallet.sol`, `CustodialWalletFactory.sol`, `NFTToken.sol`, `RushToken.sol`, `VestingWallet.sol`

Description: The contract makes use of the floating-point pragma ^0.8.7. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps to ensure that contracts are not unintentionally deployed using another compiler version, such as an obsolete version, that may introduce issues in the contract system.

Recommendation: Consider locking the pragma version. It is advised that floating pragma not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Update: The team has fixed the issue by specifying a single solidity version.

QSP-15 Using `transfer` To Send Ether Might Revert

Severity: Low Risk

Status: Fixed

File(s) affected: `CustodialWallet.sol`

Description: Usage of `address.transfer` is discouraged since it only sends 2300 gas and might revert for some fallback functions. Refer to SWC-134 for details.

Recommendation: Replace `transfer` with a low-level `call`.

Update: The team now uses `Address.sendValue` instead of the `transfer` function.

QSP-16 Approve Race

Severity: Informational

Status: Acknowledged

File(s) affected: `RushToken.sol`

Description: The standard ERC20 implementation contains a widely-known race condition in its `approve` function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using `transferFrom` to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Recommendation: Use `increaseAllowance` and `decreaseAllowance` functions to modify the approval amount instead of using the `approve` function to modify it.

Update: The team noted that they will ensure the safer `increaseAllowance` and `decreaseAllowanse` wil be used.

QSP-17 `getBeneficiaries` May Run Out Of Gas

Severity: Informational

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`

Description: The `getBeneficiaries` method may run out of gas if the linked list of beneficiaries is very long.

Recommendation: It is recommended to supply a version of the `getBeneficiaries` method where the caller can specify a number of maximum iterations and a version where the caller can specify a continuation index and beneficiary so that the list of beneficiaries can be queried in batches effectively.

Update: The team has fixed the issue by limiting the size of the array.

QSP-18 Duplication Of Access Control Logic

Severity: *Informational*

Status: Fixed

File(s) affected: `ContractSpenderManager.sol`

Description: The `ContractSpenderManager` implements its logic to track a set of addresses, so-called “spenders”. However, very similar logic is already available via the `AccessControl` contract from the OpenZeppelin library. Unless there are specific reasons for the re-implementation, a library should always be used for as much of a project's logic as possible.

Recommendation: The `ContractSpenderManager`'s logic should be replaced with the inheritance from `AccessControl` and configuration of a basic “spender” role. A similar `isSpender` method can be implemented to allow easy querying whether a specific address has the spender role.

Update: The team has fixed the issue by using the `AccessControlEnumrable` library.

QSP-19 Token ID Not Human Legible In Revert String

Severity: *Informational*

Status: Fixed

File(s) affected: `NFTChildToken.sol`

Description: The revert string in L45 combines the main error message with the token ID that is being checked. However simply packing a `uint256` value with `abi.encodePacked` will lead to the resulting string containing a 0-byte padded 32 character string added to it with the individual bytes of the token ID interpreted as characters.

Recommendation: To make the token ID not be padded and human legible it is recommended to use OpenZeppelin's `Strings.toString` helper function which converts a `uint256` value to its legible string representation. This value can then be packed together with the start error message. Due to the `abi.encodePacked` and `Strings.toString` methods using a significant amount of gas it is further recommended that the `require` invocation be transformed to an `if (condition) revert(packedErrorMessage);` structure to ensure that the formatting and packing is only done when required. `Strings.toHexString` can be used if a hexadecimal representation of the token ID is preferred.

Update: The team has fixed the issue and the `token_id` is no longer part of the require message.

QSP-20 Comments Left In the Code

Severity: *Informational*

Status: Fixed

File(s) affected: `ContractUnlockTokenManager.sol`

Description: The `getTokensToBeUnlocked` function, located in the `ContractUnlockTokenManager` contract, contains code that is completely commented out--see L60. This is currently dead code.

Recommendation: Remove the commented out code as it has no purpose.

Update: The team has removed the comments.

QSP-21 Replace Custom Linked List Implementation With Library

Severity: *Informational*

Status: Fixed

File(s) affected: `ContractTokenUnlockManager.sol`

Description: The `ContractTokenUnlockManager` currently implements its own linked list mechanism using a mapping to track the beneficiaries. While the implementation appears to be correct, the code would be significantly easier to understand and provide a cleaner API (e.g. `removeLockedAmountConfig`) if it were to be replaced by OpenZeppelin's EnumerableSet.

Recommendation: Replace the custom linked list implementation with OpenZeppelin's `EnumerableSet.AddressSet`.

Update: The custom linked list implementation has been replaced with OZ's `EnumberableSet.AddressSet`.

QSP-22 Multi Recipient ERC1155 Transfer Method May Not Work

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `contracts/CompanyCustodialWallet.sol`

Description: The `safeBatchTransferMultipleRecipients` method calls the `safeBatchTransferFrom` method for every recipient in its `recipients` array. It does so attempting to transfer the same list of token IDs and amounts. However, this method will fail if the wallet contract does not own at least `amounts[i] * recipients.length` tokens of `ids[i]`.

Recommendation: If it is intended for this method to repeatedly send the same set of tokens to all recipients this should be documented. Furthermore, balance checks should be added similar to the other methods.
However, if the `safeBatchTransferMultipleRecipients` method is intended to send different sets and amounts of tokens to different recipients then the method should be modified to accept multiple arrays of IDs and amounts.

Update: This behavior is now documented. Furthermore, no balance checks have been added, since the team argues that the additional gas required for them is too high.

QSP-23 `Rush1155Token`: Potential Proxy Implementation Not Initializable

Severity: *Undetermined*

Status: Fixed

File(s) affected: `contracts/Rush1155Token.sol`

Description: The `Rush1155Token` contract inherits from the `Initializable` contract, implying that it is meant to be a post-constructor initializable contract (as is required by proxy implementations), however no methods or modifiers from the `Initializable` library are used, and a constructor is expected to be used for initialization of the `ERC1155` logic.

Recommendation: If it was intended for the `Rush1155Token` contract to be initialized post-construction the required initialization method should be implemented. If it is not intended to be used as proxy implementation the `Initializable` library should not be referenced in the contract.

Update: The team did not intend for the contract to be a proxy and has removed the unnecessary import.

Automated Analyses

Slither

Slither reported the following :

```
*MinimalForwarder.execute(MinimalForwarder.ForwardRequest,bytes) (@openzeppelin/contracts/metatx/MinimalForwarder.sol#42-58) sends eth to arbitrary user
* ERC1967UpgradeUpgradeable._functionDelegateCall(address,bytes) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#208-214) uses delegatecall to a input-controlled function id
    - (success,returndata) = target.delegatecall(data) (@openzeppelin/contracts-upgradeable/proxy/ERC1967/ERC1967UpgradeUpgradeable.sol#212)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall
, which we classified as false positives.
```

Adherence to Specification

1.The `release` method allows any address to call it, but the documenting `README.md` file states: "Only an admin can add an unlock config to the contract for any beneficiary and release unlocked tokens to beneficiary".

Code Documentation

- The comment in `VestingWallet.sol` states that the "default implementation is a linear vesting curve" when in reality it is a vesting cliff that releases all tokens at once after the lock period has passed.
- `contracts/ContractTokenUnlockManager.sol` does not contain a single comment as to what it does or how it works. It is also not mentioned at all in the README file.

Adherence to Best Practices

1. General project settings: Lack of optimizer. It is recommended the solidity optimizer be enabled for the final compilation of contracts prior to deployment. [Unresolved]
2. General project settings: Lack of linter and fixer. It is recommended that a linter such as `solhint` be used to ensure that files are consistently formatted. Solidity code can be automatically formatted using the help of the `prettier-plugin-solidity` package, for example. [Unresolved]
3. `contracts/CustodialWalletFactory.sol`: Implicit visibility. The `beacon` and `spenderManager` properties do not have an explicit visibility. It is recommended their visibility whether `internal` / `private` / `public` be defined explicitly for the sake of transparency and clarity. [Fixed]
4. Use of OpenZeppelin `Counter` for simple counter. While there's nothing fundamentally wrong with using OpenZeppelin's `Counter`, it isn't very efficient. Making `Counter` variables into simple `uint256` variables and incrementing them in an `unchecked` context will save gas. Files where `Counter` is used : `contracts/CustodialWalletFactory.sol` (L22), `contracts/NFTToken.sol` (L14) [fixed]
5. `contracts/CustodialWalletFactory.sol`: Batching method `createWalletBatch` (L42-47) could be optimized. While simply calling `createWallet` multiple times reduces complexity, the batching of wallet creation could be made more efficient. By bringing the wallet creation logic into the `createWalletBatch` the `_walletId` variable only needs to be written once at the end of the loop. Furthermore, the constructor data `abi.encodeWithSelector(CustodialWallet.initialize.selector, spenderManager)` only needs to be encoded once and could then be reused. Another advantage is that the `onlyTransactor` modifier only needs to run once for the `createWalletBatch` method, instead of being repeatedly run for every iteration of the for-loop. [Fixed]
6. `contracts/TransactorControl.sol`: L10: Use of deprecated `_setupRole` method. According to OpenZeppelin's documentation, the `_setupRole` method is deprecated in favor of `_grantRole` (source).[Fixed]
7. `contracts/NFTChildToken.sol`: L61: `encodeTokenMetadata` method defined as external despite being used internally. If an external / public method needs to be called from within the contract defining it it should be defined as `public` in order to save gas. [Mitigated]
8. `contracts/RushToken.sol`: L35-49: The forwarder accepting `_msgSender` and `_msgData` methods are reimplemented instead of being inherited from the `ERC2771Context` contract provided by OpenZeppelin which contains identical methods. [Unresolved]
9. `contracts/VestingWallet.sol`: L21-22, L27, L29-30, L74-75, L82-83: Use of integer types smaller than 256 bits. No storage packing is being leveraged here meaning the use of integer types which are smaller than 256 bits such as `uint64` will only lead to more gas being used overall. It is recommended that the use of `uint64` be replaced with `uint256` throughout the contract. [Fixed]
10. `contracts/ContractTokenUnlockManager.sol`: Implicit library import. While OpenZeppelin's `Address` library is indirectly brought into the context of the file via the import of `SafeERC20` it is recommended that library imports be more explicit by adding an import statement. [Fixed]
11. `contracts/ContractTokenUnlockManager.sol`: Private method names `getTokensToBeUnlocked` and `isLockedAmountConfigValid` do not follow naming convention. They are defined as having `private` visibility (L64 , L96) but they don't have a leading underscore. It is recommended to keep naming conventions consistent throughout a project. The `getTokensToBeUnlocked` and `isLockedAmountConfigValid` methods should either be made public or a leading underscore should be added. [Fixed]
12. `contracts/ContractTokenUnlockManager.sol`: L96-113: Optimization possible. The output of the `isLockedAmountConfigValid` method is `false` whenever either `isAmountValid` or `isUnlockScheduleValid` become `false` due to the final logical and (L112). Instead of storing these states in two variables, the method can directly return `false` instead of first setting them to `false`. The method would return `true` if it reaches the end. Furthermore, the repeated `idx > 0` check (L105) can be removed if the for-loop runs for `lockedAmounts.length - 1` iterations, one additional `lockedAmounts[idx].amount == 0` check would have to be added. With another small optimization. [Fixed] The method body would look something like this:

```
uint256 lastIndex = lockedAmounts.length - 1;

for (uint256 idx = 0; idx < lastIndex; ) {
    if (lockedAmounts[idx].amount == 0) {
        return false;
    }// unchecked block to reduce `++idx` gas usage
unchecked {
    if (lockedAmounts[idx].unlockTime &gt; lockedAmounts[++idx].unlockTime) {
        return false;
    }
}
}
if (lockedAmounts[lastIndex].amount == 0) return false;

return true;
```

13. Event names should start with a capital letter. See `contracts/ContractTokenUnlockManager.sol` events `unlockTokenConfigAdded` and `unlockTokenConfigRemoved`. [Fixed]
14. Use `immutable` for `_distributionToken` in `contracts/ContractTokenUnlockManager.sol` since it is only changed in the constructor. [Fixed]
15. Declare `_distributionToken` as `IERC20` to avoid having to cast it on every use in `contracts/ContractTokenUnlockManager.sol`. [Fixed]
16. `contracts/CompanyCustodialWallet.sol`: Repeated checks in sub-calls. The `safeBatchTransferMultipleRecipients` and `batchTransferERC20` methods repeatedly trigger other calls in their for-loops. These methods repeat checks such as `onlySpender` or `Address.isContract` checks already checked by the parent method. Batch calling methods should refrain from calling high-level methods and instead call "lower level" methods such as the token transfer methods directly so that unused checks can be omitted. [Unresolved]
17. `contracts/NFTRootToken.sol`: Unnecessary use of `bytes` parameters. The `mint(uint256,uint256,bytes)` and `setTokenMetadata` methods both accept a `bytes` parameter, only for it to later be converted to a `string`. Due to their being no difference in how solidity treats `bytes` and `string` types, it is recommended that `string` be directly used. [Unresolved]
18. `contracts/WalletExecutor.sol`: L10-12: Reimplementation of library logic. The highlighted lines can be replaced with a call to OpenZeppelin's `Address.functionCall` as it implements the same checks and calls. [Fixed]

Test Results

Test Suite Results

npx truffle test

```
Contract: ContractUnlockTokenManager
  Deploying TokenUnlockManager
    √ Invalid distribution token (523ms)
  Adding Locked Amount Configs
    √ rejects zero address for beneficiary (189ms)
    √ rejects invalid locked amount config with zero amount (257ms)
    √ rejects invalid locked amount config with wrong unlock time (220ms)
    √ add valid locked amount config (282ms)
    √ add config from non owner (200ms)
    √ adding valid locked amount config (855ms)
    √ rejects already added beneficiary (194ms)
  Release Tokens
    √ rejects zero address for beneficiary (164ms)
    √ transferring tokens to a beneficiary who is not added yet (189ms)
    √ Release tokens for added beneficiary by owner (1869ms)
  Removing Locked Amount Configs
    √ rejects zero address for beneficiary (160ms)
    √ rejects non existent beneficiary (176ms)
    √ remove config from non owner (209ms)
    √ previous beneficiary not pointing to correct beneficiary (195ms)
    √ remove locked amount config (325ms)

Contract: CompanyCustodialWallet
  batch transfer ERC20
    √ ERC20 batch transfer using non spender (240ms)
    √ ERC20 batch transfer invalid contract address (208ms)
    √ ERC20 batch transfer no recipients (172ms)
    √ ERC20 batch transfer no amounts (267ms)
    √ ERC20 batch transfer difference in recipients and amounts (204ms)
    √ ERC20 batch transfer insufficient balance (205ms)
    √ ERC20 batch transfer via valid spender (577ms)
  batch transfer ERC1155 from
    √ ERC1155 batch transfer using non spender (282ms)
    √ ERC1155 batch transfer invalid contract address (220ms)
    √ ERC1155 batch transfer no ids (267ms)
    √ ERC1155 batch transfer no amounts (254ms)
    √ ERC1155 batch transfer using valid spender (522ms)
  batch transfer ERC1155 multiple recipients
    √ ERC1155 batch transfer using non spender (391ms)
    √ ERC1155 batch transfer invalid contract address (254ms)
    √ ERC1155 batch transfer no ids (194ms)
    √ ERC1155 batch transfer no amounts (192ms)
    √ ERC1155 batch transfer no recipients (242ms)
    √ ERC1155 batch transfer using valid spender (1023ms)

Contract: ContractSpenderManager
  Add spender
    √ Not able to add spender account from  non-owner (209ms)
    √ Not able to add non-valid address (194ms)
    √ Able to add spender account from owner (288ms)
  Remove spender
    √ Not able to remove spender account from  non-owner (227ms)
    √ Able to remove spender account from owner (239ms)

Contract: CustodialWalletFactory
  Getting beacon address from factory
    √ Non owner should not be allowed (74ms)
    √ Valid address for beacon (77ms)
  deploy invalid config
    √ Invalid initial implementation (268ms)
    √ Invalid spender implementation (267ms)
    √ Invalid nft token implementation (345ms)
  Create wallet
    √ Wallet id should start from 0 (46ms)
    √ Get current implementation (76ms)
    √ Create wallet from non-owner (142ms)
    √ Create wallet from owner (531ms)
  Create wallet in batch
    √ Create wallet in batch from non-transactor (217ms)
    √ Create wallet with zero size batch (184ms)
    √ Create wallet in batch using transactor (2014ms)
  Create wallet and mint nft in batch
    √ Create wallet and mint nft in batch  from non-transactor (201ms)
    √ Create wallet with zero size batch (235ms)
    √ Create wallet and mint nft in batch using transactor (4011ms)
  Update Wallet Implementation
    √ Update invalid implementation (184ms)
    √ Update implementation from a non owner (203ms)
    √ Update implementation from owner (976ms)

Contract: CustodialWallet
  Initializing custodial wallet
    √ Cannot be initialized with invalid contractSpender (368ms)
  Native currency
    √ balance should be 0 by default (50ms)
    √ able to send to contract (174ms)
    √ non spender shouldn't be able to withdraw money (257ms)
    √ spender shouldn't be able to withdraw more than balance (433ms)
    √ spender should be able to withdraw money (446ms)
  ERC20
    √ mint ERC20 (396ms)
    √ withDraw ERC20 using non-spender (201ms)
    √ withDraw invalid ERC20 using spender (181ms)
    √ withDraw ERC20 more than balance using spender (222ms)
    √ withDraw ERC20 using spender (639ms)
  ERC721
```

```
      √ mint ERC721 (459ms)
      √ withDraw ERC721 using non-spender (476ms)
      √ withDraw invalid ERC721 using spender (408ms)
      √ withDraw ERC721 from a non owner using spender (449ms)
      √ withDraw ERC721 from a owner using spender (690ms)
    ERC1155
      √ balance should be 0 by default (93ms)
      √ able to send to contract (328ms)
      √ non spender shouldn't be able to withdraw money (353ms)
      √ token should be a valid address (388ms)
      √ spender shouldn't be able to withdraw more than balance (474ms)
      √ spender should be able to withdraw tokens (588ms)

  Contract: ERC1155
    like an ERC1155
      balanceOf
        √ reverts when queried about the zero address (60ms)
        when accounts don't own tokens
          √ returns zero for given addresses (193ms)
        when accounts own some tokens
          √ returns the amount of tokens owned by the given addresses (189ms)
      balanceOfBatch
        √ reverts when input arrays don't match up (157ms)
        √ reverts when one of the addresses is the zero address (92ms)
        when accounts don't own tokens
          √ returns zeros for each account (97ms)
        when accounts own some tokens
          √ returns amounts owned by each account in order passed (78ms)
          √ returns multiple times the balance of the same address when asked (95ms)
      setApprovalForAll
        √ sets approval status which can be queried via isApprovedForAll (46ms)
        √ emits an ApprovalForAll log
        √ can unset approval for an operator (222ms)
        √ reverts if attempting to approve self as an operator (236ms)
      safeTransferFrom
        √ reverts when transferring more than balance (237ms)
        √ reverts when transferring to zero address (200ms)
        when called by the multiTokenHolder
          √ debits transferred balance from sender (63ms)
          √ credits transferred balance to receiver
          √ emits a TransferSingle log
          √ preserves existing balances which are not transferred by multiTokenHolder (140ms)
        when called by an operator on behalf of the multiTokenHolder
          when operator is not approved by multiTokenHolder
            √ reverts (192ms)
          when operator is approved by multiTokenHolder
            √ debits transferred balance from sender (61ms)
            √ credits transferred balance to receiver (75ms)
            √ emits a TransferSingle log
            √ preserves operator's balances not involved in the transfer (110ms)
        when sending to a valid receiver
          without data
            √ debits transferred balance from sender (94ms)
            √ credits transferred balance to receiver (74ms)
            √ emits a TransferSingle log
            √ calls onERC1155Received
          with data
            √ debits transferred balance from sender (76ms)
            √ credits transferred balance to receiver (91ms)
            √ emits a TransferSingle log
            √ calls onERC1155Received
        to a receiver contract returning unexpected value
          √ reverts (319ms)
        to a receiver contract that reverts
          √ reverts (335ms)
        to a contract that does not implement the required function
          √ reverts (395ms)
      safeBatchTransferFrom
        √ reverts when transferring amount more than any of balances (282ms)
        √ reverts when ids array length doesn't match amounts array length (424ms)
        √ reverts when transferring to zero address (186ms)
        when called by the multiTokenHolder
          √ debits transferred balances from sender (60ms)
          √ credits transferred balances to receiver (75ms)
          √ emits a TransferBatch log
        when called by an operator on behalf of the multiTokenHolder
          when operator is not approved by multiTokenHolder
            √ reverts (202ms)
          when operator is approved by multiTokenHolder
            √ debits transferred balances from sender (92ms)
            √ credits transferred balances to receiver (92ms)
            √ emits a TransferBatch log
            √ preserves operator's balances not involved in the transfer (126ms)
        when sending to a valid receiver
          without data
            √ debits transferred balances from sender (77ms)
            √ credits transferred balances to receiver (62ms)
            √ emits a TransferBatch log
            √ calls onERC1155BatchReceived
          with data
            √ debits transferred balances from sender (62ms)
            √ credits transferred balances to receiver (110ms)
            √ emits a TransferBatch log
            √ calls onERC1155Received
        to a receiver contract returning unexpected value
          √ reverts (286ms)
        to a receiver contract that reverts
          √ reverts (367ms)
        to a receiver contract that reverts only on single transfers
          √ debits transferred balances from sender (94ms)
          √ credits transferred balances to receiver (62ms)
          √ emits a TransferBatch log
          √ calls onERC1155BatchReceived
        to a contract that does not implement the required function
          √ reverts (377ms)
      Contract interface
        ERC165
          ERC165's supportsInterface(bytes4)
            √ uses less than 30k gas (204ms)
            √ claims support (109ms)
          supportsInterface(bytes4)
            √ has to be implemented
        ERC1155
          ERC165's supportsInterface(bytes4)
            √ uses less than 30k gas (244ms)
            √ claims support (77ms)
          balanceOf(address,uint256)
            √ has to be implemented
          balanceOfBatch(address[],uint256[])
            √ has to be implemented
          setApprovalForAll(address,bool)
            √ has to be implemented
          isApprovedForAll(address,address)
            √ has to be implemented
          safeTransferFrom(address,address,uint256,uint256,bytes)
            √ has to be implemented
          safeBatchTransferFrom(address,address,uint256[],uint256[],bytes)
            √ has to be implemented
    internal functions
      _mint
        √ reverts with a zero destination address (175ms)
        with minted tokens
          √ emits a TransferSingle event
          √ credits the minted amount of tokens (46ms)
      _mintBatch
        √ reverts with a zero destination address (173ms)
        √ reverts if length of inputs do not match (414ms)
        with minted batch of tokens
          √ emits a TransferBatch event
          √ credits the minted batch of tokens (79ms)
      _burn
        √ reverts when burning the zero account's tokens (205ms)
        √ reverts when burning a non-existent token id (177ms)
        √ reverts when burning more than available tokens (350ms)
        with minted-then-burnt tokens
          √ emits a TransferSingle event
          √ accounts for both minting and burning (89ms)
      _burnBatch
        √ reverts when burning the zero account's tokens (171ms)
        √ reverts if length of inputs do not match (435ms)
        √ reverts when burning a non-existent token id (188ms)
        with minted-then-burnt tokens
          √ emits a TransferBatch event
          √ accounts for both minting and burning (124ms)
    ERC1155MetadataURI
      √ emits no URI event in constructor
      √ sets the initial URI for all token types (141ms)
      _setURI
        √ emits no URI event (144ms)
        √ sets the new URI for all token types (411ms)

  Contract: ERC1155Supply
    before mint
      √ exist (79ms)
      √ totalSupply (79ms)
    after mint
```

```
      single
        √ exist (59ms)
        √ totalSupply (63ms)
      batch
        √ exist (124ms)
        √ totalSupply (150ms)
    after burn
      single
        √ exist (62ms)
        √ totalSupply (69ms)
      batch
        √ exist (124ms)
        √ totalSupply (123ms)

  Contract: ERC20Capped
    once deployed
      capped token
        √ starts with the correct cap (76ms)
        √ mints when amount is less than cap (346ms)
        √ fails to mint if the amount exceeds the cap (450ms)
        √ fails to mint after cap is reached (399ms)

  Contract: ERC721
    Contract interface
      ERC165
        ERC165's supportsInterface(bytes4)
          √ uses less than 30k gas (171ms)
          √ claims support (62ms)
        supportsInterface(bytes4)
          √ has to be implemented
      ERC721
        ERC165's supportsInterface(bytes4)
          √ uses less than 30k gas (86ms)
          √ claims support (65ms)
        balanceOf(address)
          √ has to be implemented
        ownerOf(uint256)
          √ has to be implemented
        approve(address,uint256)
          √ has to be implemented
        getApproved(uint256)
          √ has to be implemented
        setApprovalForAll(address,bool)
          √ has to be implemented
        isApprovedForAll(address,address)
          √ has to be implemented
        transferFrom(address,address,uint256)
          √ has to be implemented
        safeTransferFrom(address,address,uint256)
          √ has to be implemented
        safeTransferFrom(address,address,uint256,bytes)
          √ has to be implemented
    with minted tokens
      balanceOf
        when the given address owns some tokens
          √ returns the amount of tokens owned by the given address (90ms)
        when the given address does not own any tokens
          √ returns 0 (45ms)
        when querying the zero address
          √ throws
      ownerOf
        when the given token ID was tracked by this token
          √ returns the owner of the given token ID (76ms)
        when the given token ID was not tracked by this token
          √ reverts
      transfers
        via transferFrom
          when called by the owner
            √ transfers the ownership of the given token ID to the given address (44ms)
            √ emits a Transfer event
            √ clears the approval for the token ID (62ms)
            √ emits an Approval event
            √ adjusts owners balances (63ms)
            √ adjusts owners tokens by index (140ms)
          when called by the approved individual
            √ transfers the ownership of the given token ID to the given address (93ms)
            √ emits a Transfer event
            √ clears the approval for the token ID (63ms)
            √ emits an Approval event
            √ adjusts owners balances (45ms)
            √ adjusts owners tokens by index (156ms)
          when called by the operator
            √ transfers the ownership of the given token ID to the given address (74ms)
            √ emits a Transfer event
            √ clears the approval for the token ID (63ms)
            √ emits an Approval event
            √ adjusts owners balances (59ms)
            √ adjusts owners tokens by index (206ms)
          when called by the owner without an approved user
            √ transfers the ownership of the given token ID to the given address (77ms)
            √ emits a Transfer event
            √ clears the approval for the token ID (73ms)
            √ emits an Approval event
            √ adjusts owners balances (77ms)
            √ adjusts owners tokens by index (141ms)
          when sent to the owner
            √ keeps ownership of the token
            √ clears the approval for the token ID (59ms)
            √ emits only a transfer event
            √ keeps the owner balance (62ms)
            √ keeps same tokens by index (155ms)
          when the address of the previous owner is incorrect
            √ reverts (219ms)
          when the sender is not authorized for the token id
            √ reverts (204ms)
          when the given token ID does not exist
            √ reverts (186ms)
          when the address to transfer the token to is the zero address
            √ reverts (193ms)
        via safeTransferFrom
          with data
            to a user account
              when called by the owner
                √ transfers the ownership of the given token ID to the given address (57ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (89ms)
                √ emits an Approval event
                √ adjusts owners balances (59ms)
                √ adjusts owners tokens by index (150ms)
              when called by the approved individual
                √ transfers the ownership of the given token ID to the given address (92ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (74ms)
                √ emits an Approval event
                √ adjusts owners balances (62ms)
                √ adjusts owners tokens by index (171ms)
              when called by the operator
                √ transfers the ownership of the given token ID to the given address (94ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (62ms)
                √ emits an Approval event
                √ adjusts owners balances (108ms)
                √ adjusts owners tokens by index (157ms)
              when called by the owner without an approved user
                √ transfers the ownership of the given token ID to the given address (75ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (76ms)
                √ emits an Approval event
                √ adjusts owners balances (76ms)
                √ adjusts owners tokens by index (122ms)
              when sent to the owner
                √ keeps ownership of the token (94ms)
                √ clears the approval for the token ID (45ms)
                √ emits only a transfer event
                √ keeps the owner balance (76ms)
                √ keeps same tokens by index (173ms)
              when the address of the previous owner is incorrect
                √ reverts (155ms)
              when the sender is not authorized for the token id
                √ reverts (269ms)
              when the given token ID does not exist
                √ reverts (204ms)
              when the address to transfer the token to is the zero address
                √ reverts (189ms)
            to a valid receiver contract
              √ calls onERC721Received (295ms)
              √ calls onERC721Received from approved (224ms)
              when called by the owner
                √ transfers the ownership of the given token ID to the given address (62ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (92ms)
                √ emits an Approval event
                √ adjusts owners balances (61ms)
                √ adjusts owners tokens by index (179ms)
              when called by the approved individual
                √ transfers the ownership of the given token ID to the given address (46ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (62ms)
                √ emits an Approval event
                √ adjusts owners balances (76ms)
                √ adjusts owners tokens by index (155ms)
              when called by the operator
                √ transfers the ownership of the given token ID to the given address (104ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (91ms)
                √ emits an Approval event
                √ adjusts owners balances (78ms)
                √ adjusts owners tokens by index (170ms)
              when called by the owner without an approved user
                √ transfers the ownership of the given token ID to the given address (60ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (60ms)
                √ emits an Approval event
                √ adjusts owners balances (74ms)
                √ adjusts owners tokens by index (154ms)
              when sent to the owner
                √ keeps ownership of the token (45ms)
                √ clears the approval for the token ID (76ms)
                √ emits only a transfer event
                √ keeps the owner balance (78ms)
                √ keeps same tokens by index (142ms)
              when the address of the previous owner is incorrect
                √ reverts (212ms)
              when the sender is not authorized for the token id
                √ reverts (268ms)
              when the given token ID does not exist
                √ reverts (224ms)
              when the address to transfer the token to is the zero address
                √ reverts (178ms)
              with an invalid token id
                √ reverts (221ms)
          without data
            to a user account
              when called by the owner
                √ transfers the ownership of the given token ID to the given address (76ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (89ms)
                √ emits an Approval event
                √ adjusts owners balances (77ms)
                √ adjusts owners tokens by index (153ms)
              when called by the approved individual
                √ transfers the ownership of the given token ID to the given address (76ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (110ms)
                √ emits an Approval event
                √ adjusts owners balances (71ms)
                √ adjusts owners tokens by index (143ms)
              when called by the operator
                √ transfers the ownership of the given token ID to the given address (78ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (90ms)
                √ emits an Approval event
                √ adjusts owners balances (60ms)
                √ adjusts owners tokens by index (136ms)
              when called by the owner without an approved user
                √ transfers the ownership of the given token ID to the given address (76ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (77ms)
                √ emits an Approval event
                √ adjusts owners balances (93ms)
                √ adjusts owners tokens by index (204ms)
              when sent to the owner
                √ keeps ownership of the token (78ms)
                √ clears the approval for the token ID (76ms)
                √ emits only a transfer event
                √ keeps the owner balance (49ms)
                √ keeps same tokens by index (158ms)
              when the address of the previous owner is incorrect
                √ reverts (172ms)
              when the sender is not authorized for the token id
                √ reverts (237ms)
              when the given token ID does not exist
                √ reverts (184ms)
              when the address to transfer the token to is the zero address
                √ reverts (171ms)
            to a valid receiver contract
              √ calls onERC721Received (317ms)
              √ calls onERC721Received from approved (333ms)
              when called by the owner
                √ transfers the ownership of the given token ID to the given address (92ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (91ms)
                √ emits an Approval event
                √ adjusts owners balances (76ms)
                √ adjusts owners tokens by index (189ms)
              when called by the approved individual
                √ transfers the ownership of the given token ID to the given address (78ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (91ms)
                √ emits an Approval event
                √ adjusts owners balances (45ms)
                √ adjusts owners tokens by index (125ms)
              when called by the operator
                √ transfers the ownership of the given token ID to the given address (94ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (61ms)
                √ emits an Approval event
                √ adjusts owners balances (78ms)
                √ adjusts owners tokens by index (189ms)
              when called by the owner without an approved user
                √ transfers the ownership of the given token ID to the given address (104ms)
                √ emits a Transfer event
                √ clears the approval for the token ID (92ms)
                √ emits an Approval event
                √ adjusts owners balances (61ms)
                √ adjusts owners tokens by index (156ms)
              when sent to the owner
                √ keeps ownership of the token (109ms)
                √ clears the approval for the token ID (62ms)
                √ emits only a transfer event
                √ keeps the owner balance (77ms)
                √ keeps same tokens by index (172ms)
              when the address of the previous owner is incorrect
                √ reverts (209ms)
              when the sender is not authorized for the token id
                √ reverts (226ms)
              when the given token ID does not exist
                √ reverts (241ms)
              when the address to transfer the token to is the zero address
                √ reverts (208ms)
              with an invalid token id
                √ reverts (202ms)
          to a receiver contract returning unexpected value
            √ reverts (470ms)
          to a receiver contract that reverts with message
            √ reverts (485ms)
          to a receiver contract that reverts without message
            √ reverts (741ms)
          to a receiver contract that panics
            √ reverts (693ms)
          to a contract that does not implement the required function
            √ reverts (444ms)
      safe mint
        via safeMint
          √ calls onERC721Received — with data (511ms)
          √ calls onERC721Received — without data (458ms)
          to a receiver contract returning unexpected value
            √ reverts (639ms)
          to a receiver contract that reverts with message
            √ reverts (618ms)
          to a receiver contract that reverts without message
            √ reverts (568ms)
          to a receiver contract that panics
            √ reverts (713ms)
          to a contract that does not implement the required function
            √ reverts (415ms)
      approve
        when clearing approval
          when there was no prior approval
            √ clears approval for the token (98ms)
            √ emits an approval event
          when there was a prior approval
            √ clears approval for the token (92ms)
            √ emits an approval event
        when approving a non-zero address
          when there was no prior approval
            √ sets the approval for the target address (93ms)
            √ emits an approval event
          when there was a prior approval to the same address
            √ sets the approval for the target address (108ms)
            √ emits an approval event
          when there was a prior approval to a different address
```

```
          √ sets the approval for the target address (110ms)
          √ emits an approval event
      when the address that receives the approval is the owner
        √ reverts (238ms)
      when the sender does not own the given token ID
        √ reverts (144ms)
      when the sender is approved for the given token ID
        √ reverts (475ms)
      when the sender is an operator
        √ sets the approval for the target address (109ms)
        √ emits an approval event
      when the given token ID does not exist
        √ reverts (216ms)
    setApprovalForAll
      when the operator willing to approve is not the owner
        when there is no operator approval set by the sender
          √ approves the operator (287ms)
          √ emits an approval event (146ms)
        when the operator was set as not approved
          √ approves the operator (251ms)
          √ emits an approval event (159ms)
          √ can unset the operator approval (218ms)
        when the operator was already approved
          √ keeps the approval to the given address (268ms)
          √ emits an approval event (187ms)
      when the operator is the owner
        √ reverts (219ms)
    getApproved
      when token is not minted
        √ reverts (60ms)
      when token has been minted
        √ should return the zero address (77ms)
        when account has been approved
          √ returns approved account (140ms)
  _mint(address, uint256)
    √ reverts with a null destination address (208ms)
    with minted token
      √ emits a Transfer event
      √ creates the token (157ms)
      √ reverts when adding a token id that already exists (190ms)
  _burn
    √ reverts when burning a non-existent token id (188ms)
    with minted tokens
      with burnt token
        √ emits a Transfer event
        √ emits an Approval event
        √ deletes the token (185ms)
        √ reverts when burning a token id that has been deleted (281ms)
  Contract interface
    ERC721Metadata
      ERC165's supportsInterface(bytes4)
        √ uses less than 30k gas (206ms)
        √ claims support (108ms)
      name()
        √ has to be implemented
      symbol()
        √ has to be implemented
      tokenURI(uint256)
        √ has to be implemented
  metadata
    √ has a name (109ms)
    √ has a symbol (111ms)
    token URI
      √ return empty string by default (92ms)
      √ reverts when queried for non existent token id (92ms)
      base URI
        √ base URI can be set (331ms)
        √ base URI is added as a prefix to the token URI (379ms)
        √ token URI can be changed by changing the base URI (479ms)
  Contract interface
    ERC721Enumerable
      ERC165's supportsInterface(bytes4)
        √ uses less than 30k gas (271ms)
        √ claims support (93ms)
      totalSupply()
        √ has to be implemented
      tokenOfOwnerByIndex(address,uint256)
        √ has to be implemented
      tokenByIndex(uint256)
        √ has to be implemented
  with minted tokens
    totalSupply
      √ returns total token supply (75ms)
    tokenOfOwnerByIndex
      when the given index is lower than the amount of tokens owned by the given address
        √ returns the token ID placed at the given index (107ms)
      when the index is greater than or equal to the total tokens owned by the given address
        √ reverts (108ms)
      when the given address does not own any token
        √ reverts (76ms)
      after transferring all tokens to another user
        √ returns correct token IDs for target (268ms)
        √ returns correct token IDs for owner (173ms)
        √ returns empty collection for original owner (188ms)
    tokenByIndex
      √ returns all tokens (126ms)
      √ reverts if index is greater than supply (77ms)
      √ returns all tokens after burning token 5042 and minting new tokens (1223ms)
      √ returns all tokens after burning token 79217 and minting new tokens (1043ms)
  _mint(address, uint256)
    √ reverts with a null destination address (193ms)
    with minted token
      √ adjusts owner tokens by index (139ms)
      √ adjusts all tokens list (126ms)
  _burn
    √ reverts when burning a non-existent token id (252ms)
    with minted tokens
      with burnt token
        √ removes that token from the token list of the owner (94ms)
        √ adjusts all tokens list (123ms)
        √ burns all tokens (343ms)

Contract: ERC20
  √ has a name (78ms)
  √ has a symbol (46ms)
  √ has 18 decimals (64ms)
  total supply
    √ returns the total amount of tokens (62ms)
  balanceOf
    when the requested account has no tokens
      √ returns zero (44ms)
    when the requested account has some tokens
      √ returns the total amount of tokens (76ms)
  transfer
    when the recipient is not the zero address
      when the sender does not have enough balance
        √ reverts (205ms)
      when the sender transfers all balance
        √ transfers the requested amount (310ms)
        √ emits a transfer event (157ms)
      when the sender transfers zero tokens
        √ transfers the requested amount (333ms)
        √ emits a transfer event (138ms)
    when the recipient is the zero address
      √ reverts (217ms)
  transfer from
    when the token owner is not the zero address
      when the recipient is not the zero address
        when the spender has enough approved balance
          when the token owner has enough balance
            √ transfers the requested amount (395ms)
            √ decreases the spender allowance (238ms)
            √ emits a transfer event (175ms)
            √ emits an approval event (267ms)
          when the token owner does not have enough balance
            √ reverts (222ms)
        when the spender does not have enough approved balance
          when the token owner has enough balance
            √ reverts (237ms)
          when the token owner does not have enough balance
            √ reverts (190ms)
      when the recipient is the zero address
        √ reverts (252ms)
    when the token owner is the zero address
      √ reverts (249ms)
  approve
    when the spender is not the zero address
      when the sender has enough balance
        √ emits an approval event (158ms)
        when there was no approved amount before
          √ approves the requested amount (266ms)
        when the spender had an approved amount
          √ approves the requested amount and replaces the previous one (265ms)
      when the sender does not have enough balance
        √ emits an approval event (159ms)
        when there was no approved amount before
```

```
          √ approves the requested amount (262ms)
        when the spender had an approved amount
          √ approves the requested amount and replaces the previous one (266ms)
    when the spender is the zero address
      √ reverts (206ms)
  decrease allowance
    when the spender is not the zero address
      when the sender has enough balance
        when there was no approved amount before
          √ reverts (238ms)
        when the spender had an approved amount
          √ emits an approval event (135ms)
          √ decreases the spender allowance subtracting the requested amount (268ms)
          √ sets the allowance to zero when all allowance is removed (222ms)
          √ reverts when more than the full allowance is removed (254ms)
      when the sender does not have enough balance
        when there was no approved amount before
          √ reverts (219ms)
        when the spender had an approved amount
          √ emits an approval event (173ms)
          √ decreases the spender allowance subtracting the requested amount (283ms)
          √ sets the allowance to zero when all allowance is removed (282ms)
          √ reverts when more than the full allowance is removed (233ms)
    when the spender is the zero address
      √ reverts (205ms)
  increase allowance
    when the spender is not the zero address
      when the sender has enough balance
        √ emits an approval event (204ms)
        when there was no approved amount before
          √ approves the requested amount (252ms)
        when the spender had an approved amount
          √ increases the spender allowance adding the requested amount (237ms)
      when the sender does not have enough balance
        √ emits an approval event (172ms)
        when there was no approved amount before
          √ approves the requested amount (204ms)
        when the spender had an approved amount
          √ increases the spender allowance adding the requested amount (235ms)
    when the spender is the zero address
      √ reverts (235ms)
  _mint
    √ rejects a null account (220ms)
    for a non zero account
      √ increments totalSupply (77ms)
      √ increments recipient balance (94ms)
      √ emits Transfer event

Contract: ERC721URIStorage
  token URI
    √ it is empty by default (63ms)
    √ reverts when queried for non existent token id (75ms)
    √ can be set for a token id (204ms)
    √ reverts when setting for non existent token id (222ms)
    √ base URI can be set (236ms)
    √ base URI is added as a prefix to the token URI (456ms)
    √ token URI can be changed by changing the base URI (711ms)
    √ tokenId is appended to base URI for tokens with no URI (248ms)
    √ tokens without URI can be burnt  (359ms)
    √ tokens with URI can be burnt  (557ms)

Contract: ERC20Permit
  √ initial nonce is 0 (78ms)
  √ domain separator (83ms)
  permit
    √ accepts owner signature (269ms)
    √ rejects reused signature (373ms)
    √ rejects other signature (300ms)
    √ rejects expired permit (205ms)

Contract: HikeTokenUUPSUpgradeableMock
  √ upgrade to upgradeable implementation (575ms)
  √ upgrade to upgradeable implementation with call (872ms)
  √ upgrade to and unsafe upgradeable implementation (440ms)
  √ reject upgrade to broken upgradeable implementation (556ms)
  √ reject upgrade to non uups implementation (555ms)
  √ reject proxy address as implementation (2264ms)

Contract: ERC20Votes
  √ initial nonce is 0 (63ms)
  √ domain separator (81ms)
  √ minting restriction (224ms)
  set delegation
    call
      √ delegation with balance (971ms)
      √ delegation without balance (325ms)
    with signature
      √ accept signed delegation (575ms)
      √ rejects reused signature (436ms)
      √ rejects bad delegatee (187ms)
      √ rejects bad nonce (216ms)
      √ rejects expired permit (217ms)
  change delegation
    √ call (884ms)
  transfers
    √ no delegation (139ms)
    √ sender delegation (311ms)
    √ receiver delegation (434ms)
    √ full delegation (654ms)
  Compound test suite
    balanceOf
      √ grants to initial account (74ms)
    numCheckpoints
      √ returns the number of checkpoints for a delegate (2210ms)
    getPastVotes
      √ reverts if block number >= current block (60ms)
      √ returns 0 if there are no checkpoints (76ms)
      √ returns the latest block if >= last checkpoint block (540ms)
      √ returns zero if < first checkpoint block (584ms)
      √ generally returns the voting balance at the appropriate checkpoint (2209ms)
  getPastTotalSupply
    √ reverts if block number >= current block (76ms)
    √ returns 0 if there are no checkpoints (75ms)
    √ returns the latest block if >= last checkpoint block (587ms)
    √ returns zero if < first checkpoint block (602ms)
    √ generally returns the voting balance at the appropriate checkpoint (1298ms)

Contract: ERC2771Context
  √ not able to set forwader other than owner (221ms)
  √ recognize trusted forwarder (81ms)
  when called directly
    msgSender
      √ returns the transaction sender when called from an EOA (162ms)
      √ returns the transaction sender when from another contract (198ms)
    msgData
      √ returns the transaction data when called from an EOA (132ms)
      √ returns the transaction sender when from another contract (224ms)
  when receiving a relayed call
    msgSender
      √ returns the relayed transaction original sender (493ms)
    msgData
      √ returns the relayed transaction original data (356ms)

Contract: ERC721Root
  Mint tokens
    √ mint tokens from non predicate
    √ mint tokens from predicate (381ms)
  Mint tokens with metadata
    √ mint tokens from non predicate
    √ mint tokens from predicate (391ms)

Contract: NFTChildToken
  Safe mint
    √ only transactor can safe mint tokens (174ms)
    √ safe minting to happen in autoincrement number (283ms)
    √ only transactor can safe batch mint tokens (203ms)
    √ Safe batch mint can only for more than 0 addreses (234ms)
    √ safe minting to happen in batch (1880ms)
  Should mint token on deposit
    √ ChildChainManagerProxy can make deposit tx (397ms)
    √ Deposit called by non depositor account (267ms)
  Should burn token on withdraw
    √ Should not allow to withdraw token not owner by user (601ms)
    √ Should burn token on withdraw (598ms)
    √ Should burn token on withdraw for second time (1091ms)
  Should mint tokens on batch deposit
    √ ChildChainManagerProxy can make batch deposit tx (809ms)
  Should burn tokens on batch withdraw
    √ should not allow batch withdraw more than batchSize (220ms)
    √ Should not allow to withdraw token not owner by user (606ms)
    √ User should be allowed to withdraw in batch (1256ms)
  Withdraw tokens with metadata
    √ Should not allow to withdraw token not owner by user (541ms)
    √ Should emit event with token metadata (791ms)

Contract: Rush1155RootToken
```

```
    mint token
      √ non transactor cannot mint new tokenIds (270ms)
      √ transactor should be able to mint new tokens (328ms)
    mint token batch
      √ non transactor cannot mint new tokenIds (302ms)
      √ transactor should be able to mint new tokens (346ms)

  Contract: ERC721
    Contract interface
      AccessControlEnumerable
        ERC165's supportsInterface(bytes4)
          √ uses less than 30k gas (157ms)
          √ claims support (66ms)
        getRoleMember(bytes32,uint256)
          √ has to be implemented
        getRoleMemberCount(bytes32)
          √ has to be implemented
      AccessControl
        ERC165's supportsInterface(bytes4)
          √ uses less than 30k gas (129ms)
          √ claims support (81ms)
        hasRole(bytes32,address)
          √ has to be implemented
        getRoleAdmin(bytes32)
          √ has to be implemented
        grantRole(bytes32,address)
          √ has to be implemented
        revokeRole(bytes32,address)
          √ has to be implemented
        renounceRole(bytes32,address)
          √ has to be implemented
    base URI
      √ only owner can set base URI (316ms)
      √ only transactor should be able to set token URI (175ms)

  Contract: Rush1155ChildToken
    token uri
      √ only owner can update (299ms)
      √ owner should be able to update (255ms)
    mint tokens
      √ non transactor cannot mint new tokenIds (192ms)
      √ minting new tokenIds to happen in autoincrement number (281ms)
      √ non transactor cannot mint existing tokenIds (471ms)
      √ minting non existing tokenId (190ms)
      √ minting existing tokenIds (611ms)
      √ non transactor cannot get tokens of owners (378ms)
      √ transactor can get tokens of owners in batches (717ms)
      √ transactor can get tokens of owners in single batch (701ms)
    Should mint token on deposit
      √ ChildChainManagerProxy can make deposit tx (504ms)
      √ Deposit called by non depositor account (268ms)
      √ Deposit called on invalid user account (173ms)
    Should burn token on single withdraw
      √ Should not allow to withdraw token not owner by user (537ms)
      √ Should burn token on single withdraw (552ms)
    Should burn tokens on batch withdraw
      √ Should not allow to withdraw batch token not owner by user (856ms)
      √ Should burn token on batch withdraw (976ms)

  Contract: HikeChildToken
    Update ChildChainManager
      √ Update ChildChainManager from non owner (224ms)
      √ Update ChildChainManager with invalid address (209ms)
      √ Update ChildChainManager with valid address (698ms)
    Withdraw
      √ Withdraw tokens of user (320ms)

  Contract: TransactorRole
    with token deployed
      Only transactor function
        √ Non transactor should not be allowed (79ms)
      isTransactor
        √ should return false for non transactors (51ms)
      addTransactor
        √ should not allow invalid address (209ms)
        √ non owner cannot call (315ms)
        √ should only be called by owner (425ms)
      removeTransactor
        √ non owner cannot call (290ms)
        √ should only be called by owner (605ms)

  Contract: TransactorRoleUpgradeable
    with token deployed
      Only transactor function
        √ Non transactor should not be allowed (76ms)
      isTransactor
        √ should return false for non transactors (49ms)
      addTransactor
        √ should not allow invalid address (160ms)
        √ non owner cannot call (267ms)
        √ should only be called by owner (419ms)
      removeTransactor
        √ non owner cannot call (310ms)
        √ should only be called by owner (537ms)

  Contract: VestingWallet
    √ rejects zero address for beneficiary (222ms)
    √ check vesting contract (154ms)
    vesting schedule
      ERC20 vesting
        √ check vesting schedule (62ms)
        √ execute vesting schedule (709ms)

  Contract: Wallet Executor
    Calling executor method of WalletExecutor
      √ Calling executor method with invalid transactor (195ms)
      √ Calling executor method with invalid wallet address (229ms)
      √ Calling executor method with invalid transactionId (155ms)
      √ Calling executor method without adding wallet executor as spender (210ms)
      √ Calling executor method with valid transactor (384ms)


  646 passing (12m)
```

Code Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts\	100	100	100	100	
CompanyCustodialWallet.sol	100	100	100	100	
ContractSpendable.sol	100	100	100	100	
ContractSpenderManager.sol	100	100	100	100	
ContractTokenUnlockManager.sol	100	100	100	100	
CustodialWallet.sol	100	100	100	100	
CustodialWalletFactory.sol	100	100	100	100	
MinimalForwarder.sol	100	100	100	100	
NFTChildToken.sol	100	100	100	100	
NFTRootToken.sol	100	100	100	100	
NFTToken.sol	100	100	100	100	
Rush1155ChildToken.sol	100	100	100	100	
Rush1155RootToken.sol	100	100	100	100	
Rush1155Token.sol	100	100	100	100	
RushChildToken.sol	100	100	100	100	
RushToken.sol	100	100	100	100	
TransactorRoleControl.sol	100	100	100	100	
TransactorRoleControlUpgradeable.sol	100	100	100	100	
VestingWallet.sol	100	100	100	100	
WalletExecutor.sol	100	100	100	100	
contracts\mocks\	100	100	100	100	
CustodialWalletMock.sol	100	100	100	100	
ERC1155ReceiverMock.sol	100	100	100	100	
ERC721RecieverMock.sol	100	100	100	100	
Hike1155Mock.sol	100	100	100	100	
Hike721Mock.sol	100	100	100	100	
HikeTokenMock.sol	100	100	100	100	
Rush1155TokenMock.sol	100	100	100	100	
TransactorRoleMock.sol	100	100	100	100	
All files	**100**	**100**	**100**	**100**	

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

```
98455a5eedddaebd319b0ab028ffad29894f4998c5f4d8b0b46c8ef2a6c32fc9 ./CompanyCustodialWallet.sol
316efe75fdcbd84002085ad0073d3506fcc0344f0242aca03797b2f272a498da ./ContractSpendable.sol
b4cdbdd13b66309d2253c8b1275ce4c08d4a6393f354a786cb32b8a6b1114c4a ./ContractSpenderManager.sol
e84bea57311ce34b7857d1ec4949ffe7b0a4d0ddb5163670fd25276cd8270554 ./ContractTokenUnlockManager.sol
0e6acd788c0d99758092fcb3eac530321248febbb01c6c4e13d3ee97604ecf97 ./CustodialWallet.sol
58d26294a9a5264eea8a83444a08f7b37588ed89e677060e096301d7c875c88b ./CustodialWalletFactory.sol
306da8a66ca7ae9990bf648c677627aec1c572e331bf8749e658cdfaaa449523 ./MinimalForwarder.sol
b3bf7a48ba0aab752b0ec4b8d490bd9030a781929f740f977d5ae5bc8b58a6e5 ./NFTChildToken.sol
adbbc34f11bac41f4037493315ccf51244c3eccd54ec2279139bbcaca42eeabc ./NFTRootToken.sol
770618423ce4d609855861cb633b2adad889029163832325a41478a36766f88e ./NFTToken.sol
```

```
b10866a98c9e1de8d99d38881f626552c095a496ee4e60dc4a3a8873f5381a08 ./Rush1155ChildToken.sol
09d1d7019a1688b7ab9aa2265772732cf8a2fc809093d68f52e597373ba8e08a ./Rush1155RootToken.sol
cc8543cbe3ecc3c09d36f5da41428570bbd4b35582c4b1bbc46a8d4c5ffa6d9e ./Rush1155Token.sol
8b5d6cdf093bfd1964f182429de0678f1f4eea130da9231288e0807a0d35b68b ./RushChildToken.sol
5acd5bbb33b523d2089a547752f24b4005e4b234f044cf33f130021499b96258 ./RushToken.sol
4e8b1c2a9c89414016c3bde96ad8bf1e9f213b43c68ae570ee66fab61d6c81a6 ./TransactorRoleControl.sol
e7a7df439d23587c56bb252f7ef609ee6b7269a727e2463aa91898a70a1f931d ./TransactorRoleControlUpgradeable.sol
57ddb5e687cbe1c34953f072067494318de6bda6d143e74339d58cc4b32f345a ./VestingWallet.sol
e0dbe17d4697001c2be2a6e45cf94e2b5eeafad78eb4bfbbe1153f9c1207408b ./WalletExecutor.sol
574c57a0a44a36e6358a8ddc7780c1ae6a43909c82c0619ccfb4d5f4d25a1e40 ./CustodialWalletMock.sol
7ac0773ddeca253c5755e1aa7b20bccfa5dc01dd4cbb1ceaf9c4b9980f083282 ./ERC1155ReceiverMock.sol
cea0fedeccbe561a39114e283799df4d43b0b6d19bc92ab23691554cf0e42790 ./ERC721RecieverMock.sol
714bee7182d645ea1086b71fb2f95ea4d1e6209c4d602e6a6d6d2d0c7262d617 ./Hike1155Mock.sol
a716d0aaf21e235b2e294c479d1369e338f5d351556f107e7645406138bc5483 ./Hike721Mock.sol
0a8fe86f483b12f8f5b200eabb0c5ec298452ca740588204d340508b62863105 ./HikeTokenMock.sol
d46ed47b986d555893fdbfdc5d7e9b627c57c03817f3a4ca77d2e6cb05268c77 ./Rush1155TokenMock.sol
8ca7e15ff96775427b87c16833d98579e81e4018e1727a698920c880a3b05398 ./TransactorRoleMock.sol
f41d3292ce2ace6f4b9355778371b6005ed5d9084ef48545fabb0a8099840832 ./RushToken.sol
e37bd95888638893b341235cb25673b2537969580604355de10bb86b28fc4bdc ./contracts/ContractSpendable.sol
760a67c380b4d150947a81e351a8e727543b08cd5e597ddf5a5bfd70b91baa6b ./contracts/ContractSpenderManager.sol
3a46d5cd8721986232574ec1527febfe42042cdcc6eb91e3b72d1f313e8bad70 ./contracts/ContractTokenUnlockManager.sol
f954cd5f064787e6637effba59dfee927860ee44db6a6af746049099aad9aed9 ./contracts/CustodialWallet.sol
1bb85c10d607c6fb452a09e813db6af8f54484f058d7b4add5e5757b153e467f ./contracts/CustodialWalletFactory.sol
b62d6e62e60c0c15988e1d003b3c19623a22c92444124a669aeff4304fa9eadc ./contracts/NFTToken.sol
4e8b1c2a9c89414016c3bde96ad8bf1e9f213b43c68ae570ee66fab61d6c81a6 ./contracts/TransactorRoleControl.sol
23e69a097d8da9ed47c29d920fb0ae943da221ca4a0a7286a6481c9e9c2e5bbb ./contracts/VestingWallet.sol
```

Tests

```
420a3c3ab40dc69e52ef699a623157f6af5d7b1781ce5862868ae5559fcb9389 ./CompanyCustodialWallet.test.js
faf19f3f35a05c98ec3a0226a3b2ee0ea03eb65379e594883d47491bba2ee0a2 ./Context.behavior.js
566ff66ee886e92c32e73411329079ea6eabd013fb31da748c667342edc84c18 ./ContractSpenderManager.test.js
93bf53e8ce97b2416d6d6b4eece0eac9a80a687a8d7605cfbca012524e7f4d08 ./ContractUnlockManager.test.js
8aad85fe2b40d046f32899d9e9b3dd054b6bdbbd3a23afe7e5b8d03e56c4f785 ./CustodialWallet.test.js
ed01e582cc64c2540bdb6feda6706bdac20f2f516e897cab609ae8ab8d4fbfbf ./CustodialWalletFactory.test.js
d44afd873e69ec00fadcc3b70118598aa9e6ab7cdf2b93a7a3fe70bd65793033 ./eip712.js
e3b3f7833b82c7000ca82dafa9aedafce5367004e3da34ac3108de8a70002098 ./ERC1155.behavior.js
6c154bc3e2c0b81a49dac58d922fa0c4130adb7cfa10e620e1fb3aaf3e911827 ./ERC1155.test.js
90ee2a1b82c3488db0dbf0eef463c4bd55f5ef9bf0fe8edb422f30daa9d4e206 ./ERC1155Supply.test.js
8f781ba1c4ed75d21221ff7983071e95c01f56cc0e09a7098cd2d0b633544648 ./ERC20.behaviour.js
dbdaa7b97f89cc592c6b5f4de8be58ed51488bc98cbbbc34951ab275802aa7e8 ./ERC20Capped.behavior.js
e8a98836c2ad7890bd5f5e2777a8f2d5f062baec4132e3c134f3dcce87c4767d ./ERC20Capped.test.js
303e703891cffde377364874905363e759fc4f46bd5afc459b4eaf86b829abb7 ./ERC721.behavior.js
10da8ee2488a974f0ab83ceeb01cc9fe408af53ecbd759a3d41a83dfc1a9b362 ./ERC721.test.js
18dd65a86d77c2433236d109abdc45cda763a2201b87eabdde759df00895eea4 ./ERC721URIStorage.test.js
241e17eba132bb3cdad65fb8d64759bcf1b21e1e02414517bfe0e3d4a7f687e3 ./HikeToken.test.js
cdbfa35103d34e9f7154660d6825fb6b233773281015a2ea5d9f01224ff628aa ./HikeTokenPermit.test.js
f21f2339d59d9ddc223b741351e41bb9a88b68a053797683c44416d369e6a83e ./HikeTokenUUPSUpgradeable.test.js
19157b0ce2ef6c83101ad16e6ebd4d097f39b0235b6a28924820007bfec41776 ./HikeTokenVotes.test.js
d65e3f7c2c76131a747109e202bcc56079fe69a3d98424562f3dd629e3a3d718 ./MetaTransaction.test.js
600c23367b795776128f6add09adea93e41242f9cfdf2275fe61a89bc4053554 ./NFTChildToken.test.js
c131a21e3431454087addf95a5cfd8313466344549d33a788cf6da70bff075a1 ./NFTRootToken.test.js
db76c713df5dd014ecd1704a76a5dee1f309528a4f4576d4b69caf5866f825ce ./NFTToken.test.js
5b3ec81bad3344ed9e8c88b48fea743262fcaa748be56afd8f49d2482d5a7ffe ./Rush1155ChildToken.test.js
491929242f5065be37c623c00f69feb82100df3f5db61a82baec3c171bc64763 ./Rush1155RootToken.test.js
93d3a6cde281b4982f64a1f2da6da2cd1a0093287a03a9d8e1bdefa858adcf5f ./RushChildToken.test.js
7cd079ab6f72581453c1181885e70db2c100ad80c14494aab3d37de34c55acdb ./SupportsInterface.behavior.js
b08961c8157a2cdafb91261130985796db8b66fbc18267eb327b81e408eaf2e8 ./TransactorRole.behaviour.js
362aab89765fc123a7bd35ea34bc4deb479b33ac61eeb5bce4ebc5cab651b0b9 ./TransactorRoleControl.test.js
1b991fd38b2f13b505a8c00ebbeaa79a1cf28d6887739ffbb331b73a41ca32f3 ./TransactorRoleControlUpgradeable.test.js
47c2a5d997452a58013f85e6093c587b236fb9f532c17b8412e23071a7e76323 ./VestingWallet.behaviour.js
8a2d8930f4307fbe397be56c0e53c6387f44330fb9954a045ea15e5788961c59 ./VestingWallet.test.js
```

```
8f1ce00d998caf9cc6d3c1f01bd906a2600c8bc0355a1ac22d958aeb4f1e0311 ./WalletExecutor.test.js
420a3c3ab40dc69e52ef699a623157f6af5d7b1781ce5862868ae5559fcb9389 ./CompanyCustodialWallet.test.js
faf19f3f35a05c98ec3a0226a3b2ee0ea03eb65379e594883d47491bba2ee0a2 ./Context.behavior.js
566ff66ee886e92c32e73411329079ea6eabd013fb31da748c667342edc84c18 ./ContractSpenderManager.test.js
93bf53e8ce97b2416d6d6b4eece0eac9a80a687a8d7605cfbca012524e7f4d08 ./ContractUnlockManager.test.js
8aad85fe2b40d046f32899d9e9b3dd054b6bdbbd3a23afe7e5b8d03e56c4f785 ./CustodialWallet.test.js
ed01e582cc64c2540bdb6feda6706bdac20f2f516e897cab609ae8ab8d4fbfbf ./CustodialWalletFactory.test.js
d44afd873e69ec00fadcc3b70118598aa9e6ab7cdf2b93a7a3fe70bd65793033 ./eip712.js
e3b3f7833b82c7000ca82dafa9aedafce5367004e3da34ac3108de8a70002098 ./ERC1155.behavior.js
6c154bc3e2c0b81a49dac58d922fa0c4130adb7cfa10e620e1fb3aaf3e911827 ./ERC1155.test.js
90ee2a1b82c3488db0dbf0eef463c4bd55f5ef9bf0fe8edb422f30daa9d4e206 ./ERC1155Supply.test.js
8f781ba1c4ed75d21221ff7983071e95c01f56cc0e09a7098cd2d0b633544648 ./ERC20.behaviour.js
dbdaa7b97f89cc592c6b5f4de8be58ed51488bc98cbbbc34951ab275802aa7e8 ./ERC20Capped.behavior.js
e8a98836c2ad7890bd5f5e2777a8f2d5f062baec4132e3c134f3dcce87c4767d ./ERC20Capped.test.js
303e703891cffde377364874905363e759fc4f46bd5afc459b4eaf86b829abb7 ./ERC721.behavior.js
10da8ee2488a974f0ab83ceeb01cc9fe408af53ecbd759a3d41a83dfc1a9b362 ./ERC721.test.js
18dd65a86d77c2433236d109abdc45cda763a2201b87eabdde759df00895eea4 ./ERC721URIStorage.test.js
241e17eba132bb3cdad65fb8d64759bcf1b21e1e02414517bfe0e3d4a7f687e3 ./HikeToken.test.js
cdbfa35103d34e9f7154660d6825fb6b233773281015a2ea5d9f01224ff628aa ./HikeTokenPermit.test.js
f21f2339d59d9ddc223b741351e41bb9a88b68a053797683c44416d369e6a83e ./HikeTokenUUPSUpgradeable.test.js
19157b0ce2ef6c83101ad16e6ebd4d097f39b0235b6a28924820007bfec41776 ./HikeTokenVotes.test.js
d65e3f7c2c76131a747109e202bcc56079fe69a3d98424562f3dd629e3a3d718 ./MetaTransaction.test.js
600c23367b795776128f6add09adea93e41242f9cfdf2275fe61a89bc4053554 ./NFTChildToken.test.js
c131a21e3431454087addf95a5cfd8313466344549d33a788cf6da70bff075a1 ./NFTRootToken.test.js
db76c713df5dd014ecd1704a76a5dee1f309528a4f4576d4b69caf5866f825ce ./NFTToken.test.js
5b3ec81bad3344ed9e8c88b48fea743262fcaa748be56afd8f49d2482d5a7ffe ./Rush1155ChildToken.test.js
491929242f5065be37c623c00f69feb82100df3f5db61a82baec3c171bc64763 ./Rush1155RootToken.test.js
93d3a6cde281b4982f64a1f2da6da2cd1a0093287a03a9d8e1bdefa858adcf5f ./RushChildToken.test.js
7cd079ab6f72581453c1181885e70db2c100ad80c14494aab3d37de34c55acdb ./SupportsInterface.behavior.js
b08961c8157a2cdafb91261130985796db8b66fbc18267eb327b81e408eaf2e8 ./TransactorRole.behaviour.js
362aab89765fc123a7bd35ea34bc4deb479b33ac61eeb5bce4ebc5cab651b0b9 ./TransactorRoleControl.test.js
1b991fd38b2f13b505a8c00ebbeaa79a1cf28d6887739ffbb331b73a41ca32f3 ./TransactorRoleControlUpgradeable.test.js
47c2a5d997452a58013f85e6093c587b236fb9f532c17b8412e23071a7e76323 ./VestingWallet.behaviour.js
8a2d8930f4307fbe397be56c0e53c6387f44330fb9954a045ea15e5788961c59 ./VestingWallet.test.js
8f1ce00d998caf9cc6d3c1f01bd906a2600c8bc0355a1ac22d958aeb4f1e0311 ./WalletExecutor.test.js
98455a5eedddaebd319b0ab028ffad29894f4998c5f4d8b0b46c8ef2a6c32fc9 ./contracts/CompanyCustodialWallet.sol
e37bd95888638893b341235cb25673b2537969580604355de10bb86b28fc4bdc ./contracts/ContractSpendable.sol
760a67c380b4d150947a81e351a8e727543b08cd5e597ddf5a5bfd70b91baa6b ./contracts/ContractSpenderManager.sol
3a46d5cd8721986232574ec1527febfe42042cdcc6eb91e3b72d1f313e8bad70 ./contracts/ContractTokenUnlockManager.sol
f954cd5f064787e6637effba59dfee927860ee44db6a6af746049099aad9aed9 ./contracts/CustodialWallet.sol
1bb85c10d607c6fb452a09e813db6af8f54484f058d7b4add5e5757b153e467f ./contracts/CustodialWalletFactory.sol
306da8a66ca7ae9990bf648c677627aec1c572e331bf8749e658cdfaaa449523 ./contracts/MinimalForwarder.sol
b3bf7a48ba0aab752b0ec4b8d490bd9030a781929f740f977d5ae5bc8b58a6e5 ./contracts/NFTChildToken.sol
adbbc34f11bac41f4037493315ccf51244c3eccd54ec2279139bbcaca42eeabc ./contracts/NFTRootToken.sol
b62d6e62e60c0c15988e1d003b3c19623a22c92444124a669aeff4304fa9eadc ./contracts/NFTToken.sol
b10866a98c9e1de8d99d38881f626552c095a496ee4e60dc4a3a8873f5381a08 ./contracts/Rush1155ChildToken.sol
09d1d7019a1688b7ab9aa2265772732cf8a2fc809093d68f52e597373ba8e08a ./contracts/Rush1155RootToken.sol
cc8543cbe3ecc3c09d36f5da41428570bbd4b35582c4b1bbc46a8d4c5ffa6d9e ./contracts/Rush1155Token.sol
8b5d6cdf093bfd1964f182429de0678f1f4eea130da9231288e0807a0d35b68b ./contracts/RushChildToken.sol
f41d3292ce2ace6f4b9355778371b6005ed5d9084ef48545fabb0a8099840832 ./contracts/RushToken.sol
4e8b1c2a9c89414016c3bde96ad8bf1e9f213b43c68ae570ee66fab61d6c81a6 ./contracts/TransactorRoleControl.sol
23e69a097d8da9ed47c29d920fb0ae943da221ca4a0a7286a6481c9e9c2e5bbb ./contracts/VestingWallet.sol
574c57a0a44a36e6358a8ddc7780c1ae6a43909c82c0619ccfb4d5f4d25a1e40 ./contracts/mocks/CustodialWalletMock.sol
7ac0773ddeca253c5755e1aa7b20bccfa5dc01dd4cbb1ceaf9c4b9980f083282 ./contracts/mocks/ERC1155ReceiverMock.sol
cea0fedeccbe561a39114e283799df4d43b0b6d19bc92ab23691554cf0e42790 ./contracts/mocks/ERC721RecieverMock.sol
714bee7182d645ea1086b71fb2f95ea4d1e6209c4d602e6a6d6d2d0c7262d617 ./contracts/mocks/Hike1155Mock.sol
a716d0aaf21e235b2e294c479d1369e338f5d351556f107e7645406138bc5483 ./contracts/mocks/Hike721Mock.sol
0a8fe86f483b12f8f5b200eabb0c5ec298452ca740588204d340508b62863105 ./contracts/mocks/HikeTokenMock.sol
d46ed47b986d555893fdbfdc5d7e9b627c57c03817f3a4ca77d2e6cb05268c77 ./contracts/mocks/Rush1155TokenMock.sol
035e2df8cba5cb5b401389b1f0c744d489426dc86c9ef9b53a78a6e6329294c5 ./contracts/mocks/TransactorRoleMock.sol
```

Changelog

- 2022-03-14 - Initial report
- 2022-04-11 - Final report

About Quantstamp

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

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

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

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