Quantstamp Security Assessment Certificate

April 4th 2022 — Quantstamp Verified

Spool

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

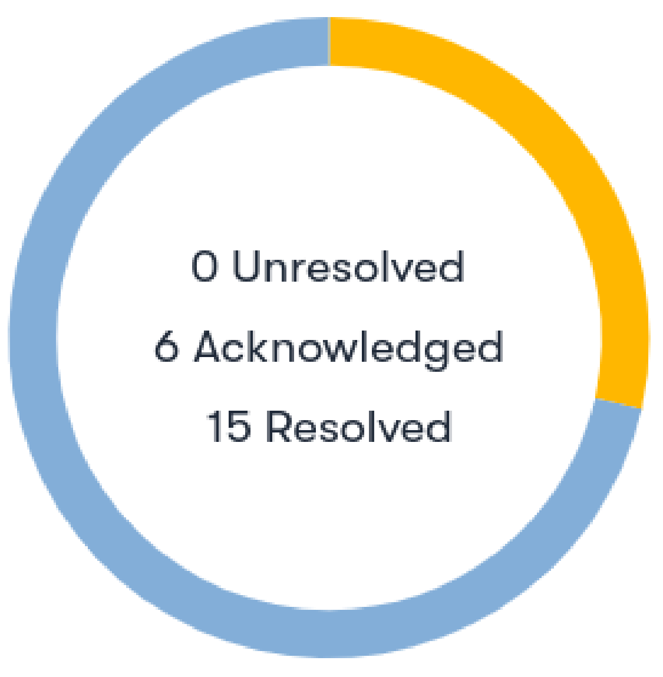
Executive Summary

Type	Permissionless Middleware
Auditors	Fayçal Lalidji, Security Auditor Cristiano Silva, Research Engineer Rabib Islam, Research Engineer
Timeline	2022-02-28 through 2022-04-04
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	User facing documentation
Documentation Quality	Medium
Test Quality	Medium

Source Code

Repository	Commit
spool-core	a9654cf

Total Issues	21	(15 Resolved)
High Risk Issues	4	(4 Resolved)
Medium Risk Issues	4	(2 Resolved)
Low Risk Issues	2	(2 Resolved)
Informational Risk Issues	10	(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 **21 potential issues** of various levels of severity. We recommend addressing the findings prior to deploying the smart contracts to the main network.
Reaudit Update
All reported issues have been either mitigated, fixed or acknowledged.

ID	Description	Severity	Status
QSP-1	Lacking Specification and Unclear Documentation	High	Mitigated
QSP-2	Attempt to Delete Items From Struct Containing Mappings	High	Fixed
QSP-3	User Might Lose Reward Due to Incorrect Update of the Instant Deposit Value	High	Fixed
QSP-4	Loss of Shares Due to Computation Precision	High	Fixed
QSP-5	Reward Notification Might Throw	Medium	Acknowledged
QSP-6	Vault Allocation Adjustment Might Be Incorrect	Medium	Acknowledged
QSP-7	Updating Reward End Distribution Does Not Update the Reward Rate	Medium	Fixed
QSP-8	Reward Withdrawal Can Be Frozen When Sent to the Recipient	Medium	Fixed
QSP-9	Review the Access Control for All External and Public Functions	Low	Fixed
QSP-10	Reentrancy Patterns	Low	Fixed
QSP-11	Missing the Emission of Important Events	Informational	Fixed
QSP-12	Is `Vault.withdrawFastWhileReallocating(...)` Really Required?	Informational	Fixed
QSP-13	Spool's Owner Can Change the Vault's Owner	Informational	Acknowledged
QSP-14	Function `Controller.getRewards(...)` Reverts the Execution in Case One Vault Is Invalid	Informational	Fixed
QSP-15	Clone-and-Own	Informational	Fixed
QSP-16	Functions That Can Be Declared External	Informational	Fixed
QSP-17	Gas Optimization: Smaller Integer Variables (`uint8`, `uint16`, ..., `uint128`) Are Not Cheaper Than `uint256` In-Memory Variables	Informational	Acknowledged
QSP-18	Contract Size Optimization: Review the Visibility of State Variables for All Contracts	Informational	Acknowledged
QSP-19	Key External Functions With Unrestricted Access	Informational	Fixed
QSP-20	Function `RiskProviderRegistry.setRisks(...)` Allows Drastic Changes in Risk Assignments	Informational	Acknowledged
QSP-21	Privileged Roles and Ownership	? 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.

DISCLAIMER: Lacking good quality inline and developer documentation impairs any independent auditing in fully understanding the underlying logic and assessing whether the code adheres to what the project specifications should be actually.

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 v0.8.0

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 Lacking Specification and Unclear Documentation

Severity: *High Risk*

Status: Mitigated

File(s) affected: `contracts/*`

Description: The project is overall underdocumented and underspecified.

- The available documentation does not clarify that the terminology in the code is not the same as the terminology that is used in the front-end (e.g. a Vault in the code is a Spool in the front-end). This may lead to major confusion for users who wish to look over the smart contract code.
- The documentation sometimes describes the functionality of the code in a circular manner (for example, the terms "redeem" and "claim" are never clearly defined).
- With a codebase so large and interconnected, there is a distinct need for documentation that can explain how all the contracts fit together and the purpose of each individual contract.
- There is no specification to describe how the system should work as a whole.

Recommendation: Documentation can be written at different levels, and should be updated while implementing the contracts: a) A plain English description of the system, describing what the contracts do and any assumptions on the codebase. b) Schema and architectural diagrams, including the contract interactions and the state machine of the system. Slither printers can help to generate these schemas. c) Thorough code documentation, the Natspec format can be used for Solidity.

Update: During the reaudit phase, NatSpec comments were added to all the contract functions. Inline comments were added. Readme was updated to provide a high-level overview of the Spool contracts. A function can be called "npm run generate-docs" which generates the markdown files containing external function and their comments. Along with the audit, some documents and diagrams were provided explaining Spool working. There are also public explainer videos. For QSP-14 a detailed document describing the roles in the Spool system was provided. Full technical documentation and official diagrams are in the process of making and will be released as soon as completed.

QSP-2 Attempt to Delete Items From Struct Containing Mappings

Severity: *High Risk*

Status: Fixed

File(s) affected: `vault/RewardDrip.sol`

Description: The function `RewardDrip.forceRemoveReward(...)` calls `delete` for a struct that contains a mapping. The mapping will not be removed and can be reused by malicious users if the removed token is added back to the contract. The code is reproduced below:

```
struct RewardConfiguration {
    uint32 rewardsDuration;
    uint32 periodFinish;
    uint192 rewardRate; // rewards per second multiplied by accuracy
    uint32 lastUpdateTime;
    uint224 rewardPerTokenStored;
    mapping(address => uint256) userRewardPerTokenPaid;
    mapping(address => uint256) rewards;
}

function forceRemoveReward(IERC20 token) external onlyOwner {
    _removeReward(token);

    delete rewardConfiguration[token];
}
```

Recommendation: Every mapping item must be deleted (set to zero) before deleting the struct entry. If not the token should be blacklisted and not relisted again.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/3

QSP-3 User Might Lose Reward Due to Incorrect Update of the Instant Deposit Value

Severity: *High Risk*

Status: Fixed

File(s) affected: `Vault.sol`

Description: Assume that a user performed a deposit toward which he got some vault shares and then performed a new deposit and executed a withdrawal for all his shares knowing that his new shares were not yet processed. His instant deposits would be set to zero even if he actually will be holding shares after the new global index is processed.

Recommendation: We recommend using a two-step update to modify the user instant deposit correctly, by only adding the new deposit value to the `User[].instantDeposit` state variable after `redeemUserModifier` is executed

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/20.

QSP-4 Loss of Shares Due to Computation Precision

Severity: *High Risk*

Status: Fixed

File(s) affected: `VaultIndexActions.sol`, `BaseStrategy.sol`

Description: In `VaultIndexActions._redeemStrategiesIndex`, `BaseStrategy._getNewSharesAfterWithdraw` and `BaseStrategy._getNewShares` the initial share value is directly proportional to the deposit amount. This functionality permits the user to set the initial share value as desired and may introduce a lower precision when calculating shares of subsequent users.
In extreme scenarios, an attacker might make an initial small deposit and manipulate the yield of a strategy through a third party which might make the precision losses even higher.

Recommendation: We recommend setting a hardcoded minimum value that will be set as shares for a user that makes a deposit that is lower than the minimum required (as example `10**18`).

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/4.

QSP-5 Reward Notification Might Throw

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `RewardDrip.sol`

Description: `RewardDrip._notifyRewardAmount()` contains the following check that might throw the notify reward transactions:

```
require(
    newRewardRate >= config.rewardRate,
    "LRR"
);
```

The check is erroneous since the leftover amount cannot be predicted exactly, and it only relies on users' interactions.

Exploit Scenario: Assuming the following successive calls
First call:

```
reward = 100
leftOver = 10
REWARD_ACCURACY = 1
rewardsDuration = 1
rewardRate1 = (100 * 1 + 10) /1=  110
```

Second call:

```
reward = 100
leftOver = 5
REWARD_ACCURACY = 1
rewardsDuration = 1
rewardRate2 = (100 * 1 + 5) /1=  105
```

In this case, `rewardRate2 < rewardRate1` means that the transaction will throw.

Recommendation: We recommend either removing or redesigning the implemented verification.

Update: The Spool team explained that the check works as intended. The system does not allow changes that would result in a lower "rewardRate" to protect users. If someone sets the emissions will last for 3 months, he cannot set it to less than that. but can only extend the period while keeping the same or higher reward rate.

QSP-6 Vault Allocation Adjustment Might Be Incorrect

Severity: *Medium Risk*

Status: Acknowledged

File(s) affected: `VaultRestricted.sol`

Description: `VaultRestricted._adjustAllocation` discards proportion redistribution for any value lower than 0.1%. However, assuming for example a vault set with five different strategies, three withdrawals of 0.1% that are discarded, one withdrawal of 0.2% that is not discarded, and one deposit of 0.5% that is not, less funds will be deposited to that specific strategy since the total optimized withdrawal value will be less than expected.

Recommendation: It should be clearly specified if the scenario above is part of the project design; otherwise, we recommend reimplementing the function as intended or discarding any value lower than 0.1% when computing the rates off-chain.

Update: The Spool team explains that ignoring small balances using a threshold of 0.1% is a business decision and is done by design.

QSP-7 Updating Reward End Distribution Does Not Update the Reward Rate

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `RewardDrip.sol`

Description: Updating the distribution end period using `RewardDrip.updatePeriodFinish(...)` can have serious consequences, either leaving the contract without enough balance to distribute the reward or leaving extra reward undistributed in the contract. This is due to not updating the reward rate according to the new set finish period.

Recommendation: The implementation of such function should clearly be associated and documented with the issues that can be created by its execution.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/5.

QSP-8 Reward Withdrawal Can Be Frozen When Sent to the Recipient

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `RewardDrip.sol`

Description: `RewardDrip.getRewards(...)` allows any user to transfer rewards to a specific account including contract addresses. Depending on the implemented logic the reward token might be frozen in the recipient contract if it does not implement a sweep function or other.

Recommendation: We recommend allowing only the `msg.sender` to withdraw its own reward balance.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/6.

QSP-9 Review the Access Control for All External and Public Functions

Severity: *Low Risk*

Status: Fixed

File(s) affected: `contracts/*`

Description: Several external/public functions are not implementing access control. This can be dangerous and exploitable by malicious users. Below we present a non-exhaustive list of functions that could benefit from more rigorous access control.

- `FeeHandler.collectEcosystemFees(...)`: not clear who is the actor supposed to call this function. However, since this is an admin task, it should implement access restrictions.
- `FeeHandler.collectTreasuryFees(...)`: not clear who is the actor supposed to call this function. However, since this is an admin task, it should implement access restrictions.

Recommendation: Check if the above proposal is consistent. In case you agree, please reflect these changes in the code.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/15.

QSP-10 Reentrancy Patterns

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Vault.sol`

Description: In the case of malicious users calling contracts and/or ERC777 transfers, certain patterns can be exploited for reentrancy attacks.
File affected:

- `Vault.sol`

Recommendation: Swap l495 with l482-493 of `Vault.sol`.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/7.

QSP-11 Missing the Emission of Important Events

Severity: Informational

Status: Fixed

File(s) affected: `contracts/*`

Description: The following functions can log events to improve the management of Spool. Events must also be triggered during contract deployment and initialization of proxied contracts.

```
- Controller.getRewards(...)
- Controller.transferToSpool(...)
- Controller.removeStrategyAndWithdraw(...)
- Controller.runDisableStrategy(...)
- Controller.runDisableStrategy(...)
- vault/VaultBase.transferVaultOwner(...)
- vault/VaultBase.lowerVaultFee(...)
- vault/VaultBase.updateName(...)
```

Recommendation: Emit events for critical parameters changes, so one can monitor the application after the deployment on Mainnet. Note that the list of functions presented here is not exhaustive.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/14.

QSP-12 Is `Vault.withdrawFastWhileReallocating(...)` Really Required?

Severity: Informational

Status: Fixed

File(s) affected: `Vault.sol`

Description: This function allows a user to withdraw their deposited funds right away while the vault is reallocating. Since reallocation is a very tricky functionality, maybe this functionality can be suppressed for now. We would like to discuss the pros and cons of not enabling this feature in the system in order to reduce the chances of misbehaviors in the reallocation process.

Recommendation: The development team must discuss the pros and cons of keeping this functionality in the system. At this moment, we consider that such functionality can be removed without any significant impact on the usability of Spool.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/1.

QSP-13 Spool's Owner Can Change the Vault's Owner

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/vault/VaultBase.sol`

Description: The Vault's owner can be changed by the Spool's owner. The documentation must be clear about the conditions that can trigger such action. Vaults can be created by end-users, and owners of vaults are compensated with fees.

```
    function transferVaultOwner(address _vaultOwner) external onlyVaultOwnerOrSpoolOwner {
        vaultOwner = _vaultOwner;
    }
```

Recommendation: Clearly state (in the documentation) the conditions in which the Spool's owner can change the Vault's owner.

Update: The Spool team explains that the Spool owner is the Spool DAO, by design it can transfer vault ownership to another address. There are multiple reasons for this. Most importantly to remove a bad actor as a Vault owner (e.g. vault owner could add vault incentive rewards that revered, effectively bricking the vault).

QSP-14 Function `Controller.getRewards(...)` Reverts the Execution in Case One Vault Is Invalid

Severity: Informational

Status: Fixed

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

Description: Function `Controller.getRewards(...)` allows a user to claim their rewards across multiple vaults in a single transaction. However, the function reverts if any of the vaults is invalid. The code is presented below:

```
    function getRewards(IVault[] calldata vaults) external {
        for (uint256 i = 0; i < vaults.length; i++) {
            require(
                validVault[address(vaults[i])],
                "Controller::getRewards: Invalid vault specified"
            );
            vaults[i].getActiveRewards(msg.sender);
        }
    }
```

Recommendation: Reverting the whole operation will lock the funds of users presenting invalid vaults. Instead of reverting the whole operation, Spool can unlock the funds of valid vaults while skipping over and logging the information of invalid vaults. Log the information of invalid vaults. Clearly state in the documentation what can lead to invalid vaults.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/8.

QSP-15 Clone-and-Own

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/vault/ReentrancyGuard.sol`

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

Recommendation: Rather than the clone-and-own approach, good industry practice is to use the Truffle framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as using libraries. Note that the file `vault/RewardDrip.sol` has the line `import "./ReentrancyGuard.sol"` instead of `import "./external/@openzeppelin/security/ReentrancyGuard.sol"`. Please, ensure that all required files use the external import instead of the cloned `ReentrancyGuard.sol`.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/9.

QSP-16 Functions That Can Be Declared External

Severity: *Informational*

Status: Fixed

File(s) affected: `contracts/*`

Description: The following functions can be declared external to reducing gas consumption.

```
- RiskProviderRegistry.setRisks(address[],uint8[])
- RiskProviderRegistry.setRisk(address,uint8)
- SpoolOwner.renounceOwnership() (contracts/SpoolOwner.sol#40-42)
- Ownable.transferOwnership(address)
```

Recommendation: The Spool team must double-check if such functions can be really made `external`.

Update: Fixed in https://github.com/SpoolFi/spool-core/pull/10.

QSP-17 Gas Optimization: Smaller Integer Variables (`uint8`, `uint16`, ..., `uint128`) Are Not Cheaper Than `uint256` In-Memory Variables

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/*`

Description: Some pieces of code use `uint8`, `uint16`, ..., `uint128` variables. However, the EVM only operates on 32 bytes / 256 bits at a time. This means that if one uses `uint8`, EVM has to first convert the `uint256` to work on it, and the conversion costs extra gas.

Recommendation: For gas optimization, consider using `uint256`. The exception is when one intends to pack variables.

Update: The Spool team provides the following response: we extensively use variable packing wherever we recognize an opportunity to optimize gas usage, as our team is most comfortable with this approach. If we read a variable as less than 256 bits in memory it's because we'll assign it to storage later. We usually do some operations in between, sometimes this serves as an overflow check. If now this would have to be done manually and cast back manually as well. It is our opinion that changing the function signatures and return statements for better gas optimization at this point would require too many resources and would result in setting back the audit considerably as well.

QSP-18 Contract Size Optimization: Review the Visibility of State Variables for All Contracts

Severity: *Informational*

Status: Acknowledged

File(s) affected: `contracts/*`

Description: The proper declaration of the visibility of state variables is a good programming practice, and it also reduces the final contract size. Public variables increase the contract size: for each public variable, the Solidity compiler automatically generates the `getter` function (https://docs.soliditylang.org/en/v0.6.10/contracts.html#visibility-and-getters). We notice the vast majority of state variables as `public` along with the whole project.

Recommendation: Review the visibility of all state variables, and keep as `public` only those variables that are really necessary. For the `public` state variables, do not provide a `getter` function.

Update: The Spool team presents the following update: "While we do agree not all the properties need to be exposed publicly, a lot of them are part of the public struct that requires to be exposed. It was a bigger issue before as we were very tight with the contract size maximum size. To address that issue it'd require significant codebase changes. Current visibility cannot harm the system but only allow for more data to be visible, so we decided to leave it as it is".

QSP-19 Key External Functions With Unrestricted Access

Severity: *Informational*

Status: Fixed

Description: Certain external functions with important roles in the Spool protocol have unrestricted access. Consider reviewing the following list of functions to ensure that their access is restricted only to addresses that should be privileged to use them.
The description above is made just to raise awareness; even if Spool is a permissionless protocol, we recommend reviewing all the functions listed below.

- `Vault.processLazy`
- `Vault.redeemVaultAndUser`
- `Vault.redeemVaultStrategies`
- `Vault.redeemUser`
- `BaseStrategy.process`
- `BaseStrategy.processReallocation`

- `BaseStrategy.processDeposit`
- `BaseStrategy.fastWithdraw`
- `BaseStrategy.claimRewards`
- `BaseStrategy.emergencyWithdraw`
- `BaseStrategy.disable`
- `CurveStrategy3CoinsBase.disable`
- `Curve3poolStrategy.disable`
- `FeeHandler.collectFees`
- `FeeHandler.collectEcosystemFees`
- `FeeHandler.collectTreasuryFees`
- `Controller.createVault`

Recommendation: Review the above functions and determine the appropriate levels of restriction for them.

Update: The Spool team presents the following answer: we have reviewed the listed functions and believe access restrictions are now at an appropriate level.

- "Vault.processLazy" was removed in another commit.
- User redeem functions only affect the user (vault redeem is a side-effect of it)
- Vault redeem can be called by anyone.
- Strategy contracts are called by the delegatecall, the restrictions shouldn't matter in this context.
- Collect fees functions were addressed in QSP-11.
- A vault can be created by any address by design. Some access restrictions were added in other QSP issues (QSP-10, QSP-11).

QSP-20 Function `RiskProviderRegistry.setRisks(...)` Allows Drastic Changes in Risk Assignments

Severity: *Informational*

Status: Acknowledged

File(s) affected: `RiskProviderRegistry.sol`

Description: The risk assignment does not make any validation. Thus, risk assignments can drastically change, making the distribution of funds across the strategies change quickly, allowing a possible ongoing hack to exploit the strategies' or vaults| funds.

Recommendation: The transition of risk assignment can be smoothed by using an arithmetic moving average where the past risk assignments are considered, but weighted over time. The drawback is that implementing this arithmetic moving average on-chain can be costly and increase the complexity of the code.

Update: The Spool team provides the following explanation: "Intended behavior. Risk providers are whitelisted by the DAO as such they are trusted entities. While we do agree that a moving average would prevent "errors", we want to make it possible for a risk provider to mark a strategy from 8 to 2, as some events (e.g. unstable USDT) might happen. These scores do not directly affect the on-chain logic but are used off-chain to calculate the vault allocations. If a risk provider starts acting weird it can always be removed by the DAO".

QSP-21 Privileged Roles and Ownership

Severity: *Undetermined*

Status: Fixed

Description: Certain contracts have state variables, e.g. `_spoolOwner`, which provide certain addresses with privileged roles. Such roles may pose a risk to end-users.
`Controller.sol` contains the following privileged roles:

- `_spoolOwnable`
  - Can add strategies by calling `addStrategy`
  - Can call `setEmergencyWithdrawer`
  - Can call `setEmergencyRecipient`
- `emergencyRecipient`
  - Receives funds when `removeStrategyAndWithdraw` or `emergencyWithdraw` are called
- `isEmergencyWithdrawer`
  - Can call `removeStrategyAndWithdraw`
  - Can call `removeStrategy`
  - Can call `emergencyWithdraw`
  - Can call `runDisableStrategy`

`SpoolReallocation.sol` contains the following privileged roles:

- `isAllocationProvider`
  - Can call reallocateVaults

`FeeHandler.sol` contains the following privileged roles:

- `_spoolOwnable`
  - Can call `setEcosystemFee`
  - Can call `setTreasuryFee`
  - Can call `setEcosystemCollector`

-
    - Can call `setTreasuryCollector`

`RiskProviderRegistry.sol` contains the following privileged roles:

- `_spoolOwnable`
    - Can call `addProvider`
    - Can call `removeProvider`

`SpoolBase.sol` contains the following privileged roles:

- `_spoolOwner`
    - Can call `setAllocationProvider`
    - Can call `setDoHardWorker`
    - Can call `setForceOneTxDoHardWork`
    - Can call `setLogReallocationProportions`
    - Can call `setAwaitingEmergencyWithdraw`

`RewardDrip.sol` contains the following privileged roles:

- `_spoolOwner`
    - Can call `updatePeriodFinish`
    - Can call `claimFinishedRewards`
    - Can call `forceRemoveReward`
- `_vaultOwner`
    - Can call `addToken`
    - Can call `notifyRewardAmount`
    - Can call `setRewardsDuration`
    - Can call `removeReward`

`VaultBase.sol` contains the following privileged roles:

- `_spoolOwner`
    - Can call `updateName`
    - Can call `transferVaultOwner`
- `_vaultOwner`
    - Can call `transferVaultOwner`

Files affected:

- `Spool.sol`
- `SpoolBase.sol`
- `Vault.sol`
- `VaultBase.sol`
- `Controller.sol`
- `FeeHandler.sol`
- `RiskProviderRegistry.sol`

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

Update: The document describing roles and their permitted actions in the Spool system was provided during the reaudit phase.

Automated Analyses

Slither

Slither did not return any significant result.

Adherence to Specification

During the first audit stage: Lack of proper documentation and functional requirements make the code hard to be read. Pieces of code related to one logical functionality are split among several files (such as the `vault` folder). The interfaces are incomplete. There are two instances of `ReentrancyGuard.sol`, and modifiers with different names making the same actions. The relation between contracts is also difficult to follow.

Code Documentation

Superficial high-level description of the system. Missing diagrams, low ratio of inline comments, no technical development documentation.

Adherence to Best Practices

BP 1: [ACKNOWLEDGED] Fees split among two or more contracts

The system has four types of fees:

1. Ecosystem fee: Circles back in Spool ecosystem and is distributed to Spool system participants;
2. Treasury fee: Collected by the Spool DAO to support the development of Spool;
3. Risk provider fee: Collected by the risk provider the vault is using to allocate it's funds;
4. Vault owner fee: Collected by the vault owner.

The functionality of managing these fees is split among (at least) two distinct contracts: a) `FeeHandler.sol`; b) `Controller.sol`. This is not a good programming practice. The unclear definition of modules, the functional requirements of each module, and their interfaces compromise the ability of developers to code, and the ability of auditors to capture logical bugs and possible exploits.
Being more specific, we understand that `FeeHandler.sol` is the contract responsible for managing fees. However, this contract only manages 3 out of 4 fees (Ecosystem, Treasury, Risk Provider). This is clear when we check the `constants` defined in the preamble of this file:

```
    /// @notice Maximum Ecosystem Fee (20%)
    uint256 public constant MAX_ECOSYSTEM_FEE = 20_00;
    /// @notice Max Treasury Fee (10%)
    uint256 public constant MAX_TREASURY_FEE = 10_00;
    /// @notice Max Risk Provider Fee (5%)
    uint256 public constant MAX_RISK_PROVIDER_FEE = 5_00;
```

On the other hand, the `Controller.sol` defines complementary constants for managing the Fees functionality, as shown below:

```
    /// @notice Maximum vault creator fee - 20%
    uint256 public constant MAX_VAULT_CREATOR_FEE = 20_00;

    /// @notice Maximum vault creator fee if the creator is the Spool DAO - 60%
    uint256 public constant MAX_DAO_VAULT_CREATOR_FEE = 60_00;
```

Hence, we are left with two contracts addressing the fees management. Furthermore, there is also a third contract that seems underutilized named `Constants.sol`, where we expect that all constants would be defined. Thus, the concept of modules is not employed properly in the application. Modules are supposed to:

1. accomplish a clear functionality in the system, so they can be easily tested;
2. not overlap the functionality of other modules: the overlap becomes a gray line between modules, creating a place that may be open to exploitation by malicious users.

BP 2: [FIXED] Contracts are not pausable

Description: A common strategy used by many smart contracts is to grant the administrator (DAO) the role to pause the system when critical situations are detected (hacking, attacks, and other malicious activities). Moreover, in case of a hack, the logging must have all the information required to trace back the balance of accounts and what led to the hack. Recommendation: Consider making all contracts pausable by using https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/Pausable.sol.

BP 3: [FIXED] `redistibutionIndex` should be named `redistributionIndex` in `VaultBase.sol`, `VaultIndexActions.sol`, and `VaultRestricted.sol`.

BP 4: [FIXED] It appears that redistribution and reallocation refer to the same thing; in this case, the same word should be used in all occurrences in the code.

BP 5: [FIXED] The word "received" is misspelled as "recieved" in `BaseStorage.sol`, `SwapHelper.sol`, `SpoolDoHardWork.sol`, `SpoolExternal.sol`, `SpoolStrategy.sol`, `BaseStrategy.sol`, `ProcessStrategy.sol`, and `RewardStrategy.sol`.

BP 6: [FIXED] There are numerous other spelling/grammatical errors in the documentation which, in addition to the relative paucity of the documentation, can make understanding the code difficult.

Test Results

Test Suite Results

```
Generating typings for: 152 artifacts in dir: build/types for target: ethers-v5
Successfully generated 266 typings!
Compiled 153 Solidity files successfully

  Controller
    contract setup tests
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
       ✓ should deploy the Controller properly
       ✓ Should fail deploying the controller with Risk Provider address 0 (311ms)
       ✓ Should fail deploying the controller with Spool address 0
       ✓ Should fail deploying the controller with Vault Factory address 0
    Vault creation tests
       ✓ should fail to create a vault with duplicate strategies
       ✓ should fail to create a vault with no strategies
       ✓ should fail to create a vault with too many strategies
       ✓ should fail to create a vault with unsupported strategy
       ✓ should fail to create a vault with improper setup
       ✓ should fail to create a vault with unsupported currency
       ✓ should fail to create a vault with improper allocations
       ✓ should fail to create a vault with invalid owner fee (41ms)
       ✓ should fail to create a vault with invalid risk provider (42ms)
       ✓ should fail to create a vault with incorrent risk tolerance (72ms)
       ✓ should add another strategy (164ms)
       ✓ should add another strategy using calldata (180ms)
       ✓ should fail to create a vault with incorrent currency for strategy (176ms)
    Updating variables tests
       ✓ Should fail hashing strategies for wrong strategy list
    strategy addition and removal tests
       ✓ should prevent addition of an existing strategy
       ✓ should allow removal of the first strategy from the system (437ms)
       ✓ should allow removal of the 4th strategy from the system (453ms)
       ✓ should prevent removal of an inexistent strategy
    Gatekeeping tests
       ✓ Should fail doing transfer to spool from non-vault address
    Emergency withdraw update tests
       ✓ Should fail trying to update emergency recipient address from non-owner account
       ✓ Should fail trying to update emergency withdrawer address from non-owner account
       ✓ Should correctly update emergency recipient address
       ✓ Should correctly update emergency withdrawer address
    Pausable controller tests
       ✓ Should fail trying to update pauser address from non-owner account
       ✓ Should fail trying to update unpauser address from non-owner account
       ✓ Should correctly update pauser address
       ✓ Should correctly update unpauser address
       ✓ Should pause (47ms)
       ✓ Should pause by spool owner
       ✓ Should unpause (132ms)
       ✓ Should unpause by spool owner (126ms)
       ✓ Should fail trying to pause from non-approved account (99ms)
       ✓ Should fail trying to unpause from non-approved account (98ms)

  Vault Fast Withdraw
getting deployment..
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
create vault..
     ✓ Should fail deploying the FastWithdraw contract with 0 addresses (1038ms)
    Fast Withdraw
       ✓ should fast withdraw user 0 (12070ms)
       ✓ should redeem user 1 shares still left in vault (125ms)
       ✓ should fast withdraw user 1 without executing withdraw (717ms)
       ✓ user 1 should have fast withdraw strategy shares
       ✓ should fail to call gatekeeped functions
       ✓ should fail to withdraw with incorrect arguments (47ms)
       ✓ should execute fast withdraw for user 1 (5617ms)
       ✓ user 1 should not be able to fast withdraw if no fast withdraw shares
       ✓ should initialize reallocation for vault (266ms)
       ✓ user 2 should not be able to normal fast withdraw when reallocation is in progress (49ms)

  Fee Handler unit tests
    Fee Handler functionality
       ✓ Pay fees, should have correct balances (139ms)
       ✓ Pay and collect fees, should collect correct amounts (1809ms)
       ✓ Collect fees when there are none, should not collect anything (83ms)
       ✓ Pay fees when all fees are 0%, should not pay anything (168ms)
    Gatekeeping tests
       ✓ Should revert if initialized with Controller as 0 address (357ms)
       ✓ Should revert if initialized with Risk Provider Registry as 0 address
       ✓ Should revert if initialized with Ecosystem Fee Collector as 0 address
       ✓ Should revert if initialized with Treasury Fee Collecter as 0 address
       ✓ Should revert if non-vault tries to pay fees
       ✓ Should revert if non-risk provider registry tries to pay fees
       ✓ Should revert if non-owner tries to set ecosystem fees
       ✓ Should revert if non-owner tries to set treasury fees
       ✓ Should revert if non-owner tries to set ecosystem collector
       ✓ Should revert if non-owner tries to set treasury collector
       ✓ Should revert if trying to set ecosystem fee collector to 0
       ✓ Should revert if trying to set treasury fee collector to 0
       ✓ Should revert if trying to set ecosystem fees to more than 20%
       ✓ Should revert if trying to set treasury fees to more than 10%
       ✓ Should revert if trying to set risk provider fees to more than 5%

  Vault Reallocation
    Reallocation one side test
getting deployment..
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
create vault..
       ✓ should initialize reallocation for vault (1909ms)
       ✓ should revert withdrawing while reallocating
       ✓ should execute doHardWork after reallocation (4536ms)
       ✓ should claim vault shares after dhw (1173ms)
    Reallocation optimize test
getting deployment..
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
strat0: 0x5fc748f1FEb28d7b76fa1c6B07D8ba2d5535177c
strat1: 0xB82008565FdC7e44609fA118A4a681E92581e680
create vault 0..
create vault 1..
       ✓ should initialize reallocation for both vaults (same amounts in opposite direction) (3579ms)
       ✓ doHardWork, vaults should exchange shares, without getting hit by the fee (5517ms)
       ✓ strategies should have same amount of shares after reallocation
       ✓ should claim vault0 shares after dhw (971ms)
       ✓ should claim vault1 shares after dhw (955ms)

  Remove Strategy
    Remove strategy and withdraw
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
create vault..
       ✓ remove strategy, should revert when user had no rights
       ✓ should remove and withdraw strategy from the system (2363ms)
       ✓ Should remove strategy from the vault (854ms)
       ✓ Should deposit to vault (650ms)
       ✓ should remove and withdraw strategy from the system again (1032ms)
       ✓ Should remove last strategy from the vault (237ms)

  RewardDrip
    Rewards Drip Configuration
       ✓ Should add one token (289ms)
       ✓ Should add two tokens (866ms)
       ✓ Should not add previously force-removed token (437ms)
    Rewards Drip Emitting Rewards
       ✓ getActiveRewards should fail if not invoked by controller (50ms)
       ✓ Deposit one user, should claim rewards proportionally (261ms)
       ✓ Deposit two users, should claim rewards proportionally (521ms)
       ✓ Deposit two users, two rewards, should claim rewards proportionally (1665ms)
    Rewards Drip Emitting Rewards, deposit before rewards
       ✓ Deposit before reward start, should claim rewards proportionally (806ms)
    End of life tests
       ✓ Should be able to remove token after finish, while user can still claim the token reward (613ms)

  RiskProviderRegistry
    contract setup tests
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
       ✓ Should fail deploying the Risk Provider with address 0
       ✓ Should fail deploying the Risk Provider with address 0
    provider tests
       ✓ Should try to add the same provider again and fail
       ✓ Should try to remove an inexistent provider
       ✓ Should remove an existing provider (104ms)
    risk tests
       ✓ Should set risks using strategies for provider set (57ms)
       ✓ Should try to set risks from non-provider account and fail
       ✓ Should try to set risks with invalid risk scores
       ✓ Should try to set risks with invalid array sizes and fail

  Spool
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
create vault..
    contract setup tests
       ✓ Should fail attempting to renounce ownership on spool owner contract
       ✓ Should fail deploying the Spool with Spool address 0
       ✓ Should fail deploying the vault factory with Controller address 0
    Spool public functions
       ✓ Should get underlying with no shares
       ✓ Should get strategy underlying
       ✓ Should get strategy underlying
       ✓ Should get vault total underlying at index 0
       ✓ Should get completed global index
       ✓ Should get active global index if batch is not completed
       ✓ Should get vault strat shares
       ✓ Should not be in mid reallocation

  Strategies Unit Test: AAVE
    Deployment Gatekeeping
       ✓ Should fail deploying Aave with stkAave address 0 (290ms)
       ✓ Should fail deploying Aave with Lending Pool Addresses Provider address 0
       ✓ Should fail deploying Aave with Aave Incentives Controller address 0
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (4432ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (13658ms)
```

```
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (39981ms)
         ✓ Process withraw, should withdraw from strategy (5560ms)
mining blocks...
mined
         ✓ Claim rewards, should deposit in strategy (9908ms)
mining blocks...
mined
         ✓ Fast withdraw, should withdraw from strategy (10549ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (4558ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (2573ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (9397ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (13140ms)
         ✓ Process withraw, should withdraw from strategy (4435ms)
mining blocks...
mined
         ✓ Claim rewards, should deposit in strategy (10529ms)
mining blocks...
mined
         ✓ Fast withdraw, should withdraw from strategy (10941ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (4971ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (2404ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (9163ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (15491ms)
         ✓ Process withraw, should withdraw from strategy (5603ms)
mining blocks...
mined
         ✓ Claim rewards, should deposit in strategy (10443ms)
mining blocks...
mined
         ✓ Fast withdraw, should withdraw from strategy (10937ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (4415ms)

  Strategies Unit Test: BarnBridge multi reward
    Deployment: Gatekeeping
       ✓ Should fail deploying Barn Bridge Multi Reward Strategy with smart yeild address 0
       ✓ Should fail deploying Barn Bridge Multi Reward Strategy with smart pool address 0
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (707ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (23785ms)
barnBridgeContract.process([], false, []);
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (10210ms)
balance1 0
balance2 994856157819842066717275
balance3 994856158717919489129838
         ✓ Process withraw, should withdraw from strategy (7387ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (5390ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (7964ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (7581ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (1688ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (22072ms)
barnBridgeContract.process([], false, []);
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (9763ms)
balance1 0
balance2 994895909380
balance3 994895910338
         ✓ Process withraw, should withdraw from strategy (7939ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (5366ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (8034ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (8179ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (1340ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (21104ms)
barnBridgeContract.process([], false, []);
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (9306ms)
balance1 0
balance2 994924591661
balance3 994924592456
         ✓ Process withraw, should withdraw from strategy (7822ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (5247ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (7929ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (8052ms)

  Strategies Unit Test: Compound
    Deployment Gatekeeping
       ✓ Should fail deploying Compound Strategy with cToken address 0
       ✓ Should fail deploying Compound Strategy with comptroller address 0
       ✓ Should fail deploying Compound Strategy with the wrong strategy helper (3423ms)
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (1134ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (8444ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (11398ms)
         ✓ Process withraw, should withdraw from strategy (6589ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (6699ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (7680ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (4608ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (1669ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (8535ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (8964ms)
         ✓ Process withraw, should withdraw from strategy (4988ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (7318ms)
mining blocks...
```

```
mined
         ✓ Fast withdraw, remove shares (7469ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (5255ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (2948ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (7765ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (8904ms)
         ✓ Process withraw, should withdraw from strategy (5926ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (7693ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (7726ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (5143ms)

  Strategies Unit Test: Convex 3pool
     ✓ Should fail deploying Convex 3 Pool Strategy with underlying address 0
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (3340ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (43708ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (22290ms)
         ✓ Process withraw, should withdraw from strategy (8969ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (10925ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (11675ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (8073ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (149ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (5711ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (14562ms)
         ✓ Process withraw, should withdraw from strategy (9594ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (10540ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (25400ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (8131ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (446ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (5703ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (9894ms)
         ✓ Process withraw, should withdraw from strategy (9319ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (12212ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (9578ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (8075ms)

  Strategies Unit Test: Curve Liquidity Gauge 3pool
     ✓ Should fail deploying Curve Liquidity Gauge 3pool with pool address 0 (66ms)
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (414ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (5814ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (11770ms)
         ✓ Process withraw, should withdraw from strategy (8743ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (9428ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (9340ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (7408ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (135ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (6355ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (10969ms)
         ✓ Process withraw, should withdraw from strategy (8373ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (9050ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (9749ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (7335ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (146ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (6208ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (14101ms)
         ✓ Process withraw, should withdraw from strategy (9698ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (9909ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (10898ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (8318ms)

  Strategies Unit Test: Harvest
     ✓ Should fail deploying Harvest with token address 0
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (50ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (17371ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (9333ms)
         ✓ Process withraw, should withdraw from strategy (6293ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (5178ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (6690ms)
mining blocks...
mined
```

```
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (6149ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (79ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (15737ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (7221ms)
         ✓ Process withraw, should withdraw from strategy (6020ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (4974ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (6075ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (5839ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (46ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (16153ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (7684ms)
         ✓ Process withraw, should withdraw from strategy (7429ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (4901ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (5750ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (5597ms)

  Strategies Unit Test: Idle
     ✓ Should fail deploying Idle Strategy with underlying address 0
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (435ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (16171ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (15736ms)
         ✓ Process withraw, should withdraw from strategy (12896ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (11184ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (12643ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (9605ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (428ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (13291ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (12000ms)
         ✓ Process withraw, should withdraw from strategy (12350ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (11362ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (12307ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (9961ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (321ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (17457ms)
mining blocks...
mined
         ✓ Process deposit twice, should redeposit rewards (13223ms)
         ✓ Process withraw, should withdraw from strategy (12526ms)
mining blocks...
mined
         ✓ Claim rewards, should claim and swap rewards for the underlying currency (11621ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (11490ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (9937ms)

  Strategies Unit Test: AAVE
    Deployment Gatekeeping
       ✓ Should fail deploying MasterChef with Masterchef address 0

  Strategies Unit Test: Yearn
     ✓ Should fail deploying Yearn with vault address 0
    Gatekeeping
       ✓ Claim rewards, should throw as Yearn Vault has no rewards (969ms)
    Asset: DAI
      Deployment: DAI
         ✓ Should deploy (54ms)
      Functions: DAI
         ✓ Process deposit, should deposit in strategy (3968ms)
mining blocks...
mined
         ✓ Process deposit twice, should deposit the second time and get the same number of shares (2013ms)
         ✓ Process withraw, should withdraw from strategy (2135ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (1671ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (1674ms)
    Asset: USDC
      Deployment: USDC
         ✓ Should deploy (663ms)
      Functions: USDC
         ✓ Process deposit, should deposit in strategy (4688ms)
mining blocks...
mined
         ✓ Process deposit twice, should deposit the second time and get the same number of shares (2727ms)
         ✓ Process withraw, should withdraw from strategy (2491ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (2315ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (1807ms)
    Asset: USDT
      Deployment: USDT
         ✓ Should deploy (971ms)
      Functions: USDT
         ✓ Process deposit, should deposit in strategy (5115ms)
mining blocks...
mined
         ✓ Process deposit twice, should deposit the second time and get the same number of shares (2713ms)
         ✓ Process withraw, should withdraw from strategy (2699ms)
mining blocks...
mined
         ✓ Fast withdraw, remove shares (2153ms)
mining blocks...
mined
         ✓ Emergency withdraw, should withdraw all funds and send it to the recipient (2690ms)

  Vault
    contract setup tests
       ✓ Should fail deploying the Vault with address 0 (46ms)
    vault creation test
getting deployment..
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
        ✓ should be created properly by the Controller (2154ms)
    vault deposit / withdraw tests
        ✓ should properly mint shares for a deposit (1087ms)
        ✓ should allow for second deposit (2462ms)
        ✓ should properly mint shares for two deposits from the same user (1081ms)
        ✓ should execute doHardWork after deposit (4746ms)
        ✓ should claim vault and user shares after deposit and dhw (1200ms)
        ✓ should execute withdrawal action (1795ms)
        ✓ should execute doHardWork (4741ms)
        ✓ should redeem vault and user shares after withdrawal and dhw (2720ms)
        ✓ should claim vault and user shares (580ms)
        ✓ Should fail if controller paused (127ms)

  VaultUpdate
    Commence vault update tests
create accounts..
deploy tokens..
deploy pools..
deploy spool..
deploy strategies..
        ✓ Should fail trying to transfer the vault owner
        ✓ should fail trying to lower the vault fee with too high fee
        ✓ should fail trying to lower the vault fee from non-vault owner account
        ✓ should fail trying to update the vault name
        ✓ Should transfer the vault owner (54ms)
        ✓ should lower the vault fee (56ms)
        ✓ should update the vault name (91ms)
    Commence vault index tests
Should deposit at index x...
Should partially complete DHW for index x...
Should deposit at index x+1...
Should complete DHW for index x...
        ✓ Should Execute withdraw (2083ms)
Should deposit at index x...
Should partially complete DHW for index x...
Should deposit at index x+1...
Should complete DHW for index x...
        ✓ Should perform DHW again and then withdraw (7509ms)


  312 passing (28m)
```

Code Coverage

Quantstamp usually recommends developers to increase the branch coverage to 90%and above before a project goes live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase. For branch code coverage, the current targeted files by the audit achieve a lower score that can be improved further.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	84.1	77.84	93.94	84.52	
Controller.sol	89.09	83.93	92.5	89.08	… 703,708,709
FastWithdraw.sol	97.67	85	100	97.73	248
FeeHandler.sol	100	95	100	100	
RiskProviderRegistry.sol	100	100	100	100	
Spool.sol	100	100	100	100	
SpoolOwner.sol	100	100	100	100	
Vault.sol	63.97	43.18	83.33	64.49	… 609,612,614
contracts/interfaces/	100	100	100	100	
IBaseStrategy.sol	100	100	100	100	
ICompoundStrategyContractHelper.sol	100	100	100	100	
IController.sol	100	100	100	100	
IFastWithdraw.sol	100	100	100	100	
IFeeHandler.sol	100	100	100	100	
IRiskProviderRegistry.sol	100	100	100	100	
ISpool.sol	100	100	100	100	
ISpoolOwner.sol	100	100	100	100	
IStrategyContractHelper.sol	100	100	100	100	
ISwapData.sol	100	100	100	100	
IVault.sol	100	100	100	100	
contracts/interfaces/spool/	100	100	100	100	
ISpoolBase.sol	100	100	100	100	
ISpoolDoHardWork.sol	100	100	100	100	
ISpoolExternal.sol	100	100	100	100	
ISpoolReallocation.sol	100	100	100	100	
ISpoolStrategy.sol	**100**	**100**	**100**	**100**	
contracts/interfaces/vault/	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IRewardDrip.sol	100	100	100	100	
IVaultBase.sol	100	100	100	100	
IVaultDetails.sol	100	100	100	100	
IVaultImmutable.sol	100	100	100	100	
IVaultIndexActions.sol	100	100	100	100	
IVaultRestricted.sol	100	100	100	100	
contracts/libraries/	80	100	81.82	81.82	
Bitwise.sol	75	100	75	75	20
Hash.sol	75	100	75	75	18
Math.sol	100	100	100	100	
contracts/libraries/Max/	100	100	100	100	
128Bit.sol	100	100	100	100	
contracts/shared/	73.77	66.67	92.86	72.41	
BaseStorage.sol	100	100	100	100	
Constants.sol	100	100	100	100	
SpoolOwnable.sol	100	100	100	100	
SpoolPausable.sol	100	100	100	100	
SwapHelper.sol	69.81	60	85.71	66.67	… 235,236,239
SwapHelperMainnet.sol	100	100	100	100	
contracts/spool/	86.9	55.56	88.51	87.24	
SpoolBase.sol	90.57	65.79	91.43	91.94	… 156,210,239
SpoolDoHardWork.sol	92.98	53.23	100	93.33	… 410,461,577
SpoolExternal.sol	97.65	77.78	100	97.75	285,286
SpoolReallocation.sol	87.76	56.25	100	87.76	… ,64,150,151
SpoolStrategy.sol	61.64	32.14	69.57	59.72	… 342,343,461
contracts/strategies/	90.52	80.61	93.02	90.48	
BaseStrategy.sol	86.17	73.91	90.91	86.17	… 375,432,445
ClaimFullSingleRewardStrategy.sol	87.5	80	100	86.67	72,73
MultipleRewardStrategy.sol	100	100	100	100	
NoRewardStrategy.sol	66.67	100	80	66.67	46
ProcessStrategy.sol	97.78	93.75	100	97.78	115,116
RewardStrategy.sol	85.19	70	100	85.19	81,82,83,84
contracts/strategies/aave/	100	75	100	100	
AaveStrategy.sol	100	75	100	100	
contracts/strategies/barnBridge/	87.5	55	100	89.09	
BarnBridgeMultiRewardStrategy.sol	87.5	55	100	89.09	… 103,106,109
contracts/strategies/compound/	98.41	67.65	100	98.44	
CompoundContractHelper.sol	97.44	58.33	100	97.5	155
CompoundStrategy.sol	100	90	100	100	
contracts/strategies/convex/	84.38	52.78	90	84.54	
ConvexBoosterContractHelper.sol	88.46	58.33	85.71	88.89	104,125,126
ConvexSharedStrategy.sol	82.86	50	92.31	82.86	… 177,178,224
contracts/strategies/curve/	85.71	40	90	85.71	
Curve3poolStrategy.sol	85.71	40	90	85.71	… 121,123,131
contracts/strategies/curve/base/	**90.2**	**58.33**	**90.91**	**85.45**	
CurveStrategy3CoinsBase.sol	54.55	0	75	46.67	… 52,53,54,55

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
CurveStrategyBase.sol	100	70	100	100	
contracts/strategies/harvest/	100	66.67	100	100	
HarvestStrategy.sol	100	66.67	100	100	
contracts/strategies/idle/	92.86	66.67	100	92.59	
IdleStrategy.sol	92.86	66.67	100	92.59	167,168,171,177
contracts/strategies/masterchef/	100	100	50	100	
MasterChefStrategy.sol	100	100	0	100	
MasterChefUsdcStrategy.sol	100	100	100	100	
contracts/strategies/masterchef/shared/	90	61.11	100	90	
MasterChefMainnetStrategy.sol	100	100	100	100	
MasterChefStrategyBase.sol	90	61.11	100	90	83,84,86,87
contracts/strategies/yearn/	100	70	100	100	
YearnStrategy.sol	100	70	100	100	
contracts/vault/	92.11	72.41	93.06	92	
RewardDrip.sol	80.95	57.89	84	82.98	… 235,238,258
VaultBase.sol	95.65	68.18	96	96.15	270,272
VaultImmutable.sol	100	100	100	100	
VaultIndexActions.sol	95.51	92.11	100	95.65	187,225,227,228
VaultNonUpgradableProxy.sol	100	100	100	100	
VaultRestricted.sol	100	66.67	100	96.36	141,151
All files	**88.32**	**68.18**	**92.66**	**88.35**	

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

```
5d548d48fadbaf523da6f3408febcedcd52588f32d70acaff3084e7f5073989b ./contracts/Vault.sol
fee9adad2565647c1cd28da2de9e44ccd5d4119fe82106b145a84dc3d1445025 ./contracts/RiskProviderRegistry.sol
a1279956997b7ebd659c521d067bf55a0ff22017ff298b1f02f825d427e60b06 ./contracts/Controller.sol
885e1d05f463ef3356f318a93e2d5ddc17f3de3465be7493e33c643c82259601 ./contracts/SpoolOwner.sol
f8ded5dc40008f273e46cfb2776eca1172b1f4351b063f7be1e05767a3100506 ./contracts/FastWithdraw.sol
8d79180f410349b558d820a32f381d16791d2566785254aa2fee6c47a51d1797 ./contracts/FeeHandler.sol
0730b8ba2706111adaf2aaf3cb2a26512fe134bbada54eb495bbdd52fd6f34c8 ./contracts/Spool.sol
6039aec580577a4beed33d7d9b01260f0fe9977ae42acd2f6d8965bc7a192e16 ./contracts/spool/SpoolBase.sol
4ca72faa44532dc3386e16703a61a0ddbd603b3e4b78cb74f8e2acdc3bae12c0 ./contracts/spool/SpoolDoHardWork.sol
1251a933abe0bee50462b4b07fe85703002704aa0edee4b40937d016253b36a0 ./contracts/spool/SpoolReallocation.sol
5852f617655269e246ae573a97f0e65284e8ddcc94b7f844ea946ec914ed59ea ./contracts/spool/SpoolStrategy.sol
c08b08047b0402b9d9ea44412852a8d56af8b8d60c91b174a5c8291e9800058b ./contracts/spool/SpoolExternal.sol
bdfdae074d80a57465e3ef2ee9f15510d945dad0297cd7db0c5e4fc2531e3486 ./contracts/shared/Constants.sol
fba82731c96af9280d1b2ea5542676b30d53a7be4b276d79df099ef145813db2 ./contracts/shared/BaseStorage.sol
ae9cc2373af49357a2a79d2ba0b92073161180878939a73413d9d8c061bd865f ./contracts/shared/SwapHelperMainnet.sol
81d389cfdb2bc740e4028d5840387adf26fee8c8fb494dd0df53498774735630 ./contracts/shared/SpoolOwnable.sol
f2dd5fd821d2824f2fe8cc3cebb41180f54548e5a499ef5923e77f0a7e661c7b ./contracts/shared/SwapHelper.sol
0b25e956682ce107031fd73fe8e3cff59f98520401cf3667eaa4f181edc130c7 ./contracts/strategies/MultipleRewardStrategy.sol
975844f12b61ecad64afe22fc656c925bc28de76714ad53b744c469c6a447ffe ./contracts/strategies/ProcessStrategy.sol
92a23e0ac35942fbb2408472163eaab5be3d3076e615f0654d6addc392bed972 ./contracts/strategies/ClaimFullSingleRewardStrategy.sol
0a6bd0c696ccbc7a429209226d1063a1a8d86bdedc7ed606c32bf9efc96db274 ./contracts/strategies/BaseStrategy.sol
7cda23d243544d1ee73031c5de412a41536354925c52e997f01c3c2e7bbea2cf ./contracts/strategies/RewardStrategy.sol
2e132ddd3c23c389dac61af75473b924cf859e479a7dc9281c87e393ecf52fbc ./contracts/strategies/NoRewardStrategy.sol
331d1b313c1c2f1956c2e06aae9130e0d19e07db69d23839f066d198a05be0a7 ./contracts/strategies/idle/IdleStrategy.sol
```

```
aeb2e04f838a819ac9672c2e2e58cf506c9d620b007021b919d12372863b22e6 ./contracts/strategies/masterchef/MasterChefStrategy.sol
3e2f8692f813fdcbfc0b7aa78304182b9f18a039892addc069c7d952b1720f9d ./contracts/strategies/masterchef/MasterChefUsdcStrategy.sol
c3f34df4e72d2a35c34cc71e152c82fe924c70a24c9d587df48b1414539c1636 ./contracts/strategies/masterchef/shared/MasterChefStrategyBase.sol
0b7c271825ce78834d992b1b0ca67fd12630b6e8031e0e071a74e4fdd774ad48 ./contracts/strategies/masterchef/shared/MasterChefMainnetStrategy.sol
4513c4232badfc171714c73839490e9b1576872e99bfbdb4db9e12da0bda461d ./contracts/strategies/convex/ConvexSharedStrategy.sol
cf74b908e8a485ec5ffab8cf3a18e02db301ad398d2058212254927c1953292f ./contracts/strategies/convex/ConvexBoosterContractHelper.sol
59d7600c5594ee742c2d0d73ea636068705eb405028d62a4b919dda3cf2e9804 ./contracts/strategies/compound/CompoundStrategy.sol
e971a3afb4978a384382aeff73fc77f4b04d2a185ed776eb873733fd71d2afbd ./contracts/strategies/yearn/YearnStrategy.sol
7716f118c164ae371abe43fb89f3b4f447458892677d53d3929ec1a7d1148e30 ./contracts/strategies/aave/AaveStrategy.sol
fbd7ac078c2a8d876c6d2ea8efbdaddcacc02bd4e29607b24edeac5cbcb30eb0 ./contracts/strategies/harvest/HarvestStrategy.sol
b831bb82aee200d30f55f1e93c083591e9a259adb4d2af023cd092117ca0372d ./contracts/strategies/barnBridge/BarnBridgeMultiRewardStrategy.sol
67b767954da29ced2cf057e2237956781e3e3ef4fd49c81901aab46c3370e8fa ./contracts/strategies/curve/Curve3poolStrategy.sol
0be5d08a731702266e318c6f8d90f3b53e4a9dc56cc1a44ca1b428dde7888203 ./contracts/strategies/curve/base/CurveStrategyBase.sol
c6b6f83c5aac24eef16ef71ac3d42be3b55a9b67273cd67e1c9d12ff261ce087 ./contracts/strategies/curve/base/CurveStrategy3CoinsBase.sol
87607afa527c750a4cbd48df89296f22a19ae4196b350aec05ea3f0714111846 ./contracts/vault/RewardDrip.sol
0a65a57eb4b1bfae576f39ab5897384220a8fd6997fb3103f20bbc435832c196 ./contracts/vault/VaultImmutable.sol
6ac4ade174d03ef70a3b4fe449a3fe1adfb4bf64bf36fa50d5af30cc2842f13b ./contracts/vault/VaultBase.sol
2ff81433ebf67ef5dcd896d60a55ec6abd4961e8159c0340062072e26a07f1c9 ./contracts/vault/VaultRestricted.sol
0489a9f75edac75902461bed15c04791dd6dd1ae1149b633bead1b45963548f6 ./contracts/vault/VaultNonUpgradableProxy.sol
a74cf1e02bdd74a4009538dbcf8d7d9afbbbd4dd0583d4eeaa4d59a4ae471b30 ./contracts/vault/VaultIndexActions.sol
9cadeefe4f1b86c9c230b95ab636dcb3adbbf20e7a994fe7d8761f2a138512fe ./contracts/vault/ReentrancyGuard.sol
d450564d1da9310acb4c5af964a47e06f6be8e02ec4fdd93cfea0c647215c4b6 ./contracts/interfaces/ISpoolOwner.sol
4c22a37afa8017f9751215744284c391f14b1855182dda7040c8a219d9035814 ./contracts/interfaces/IBaseStrategy.sol
52d06e6484fb7dc0a7688ced4447e260ec6d19485a91cac2da658ff681d4e8d4 ./contracts/interfaces/IVault.sol
ca31d87ba58c649393ea4b680e165c96d0cf03ab71293d6e3cfcf56040fb40a9 ./contracts/interfaces/IFeeHandler.sol
3bc2f447ed213a725c8b4bf7643a9277a3e180e0949042eabcbd8ee83f983382 ./contracts/interfaces/ISpool.sol
2dad6ae5ee3d4685c41bc3ebffbc11043f0265ce8599a1c70e5b32f2ad136108 ./contracts/interfaces/IController.sol
56daf99289ac5b20442f90cf5012958ca2cea4f87acaf78115b02ef9f8718091 ./contracts/interfaces/IFastWithdraw.sol
cafa6d4fbde663983c605057e507483cf45b6f4d6d20ba3d950906af93b1d906 ./contracts/interfaces/ISwapData.sol
e0453b6890a071208724828a9ea3a2472e9b8ab3d1517f9359cae34e13796fc5 ./contracts/interfaces/IRiskProviderRegistry.sol
b0e19af1f337f1a15b3524fa1ef807b884418f67dda409642ce59babf4815142 ./contracts/interfaces/IStrategyContractHelper.sol
315ff8def6d869f29028c67d76ddc6e2d319e96c1052086b963b4c0fd083af84 ./contracts/interfaces/spool/ISpoolStrategy.sol
b6cc58f67f0678e912ad0b1d0843cf8ccb406d6523fee63b41bff7fe17986376 ./contracts/interfaces/spool/ISpoolBase.sol
1e45c61ec6b7eaaf2d7b8916ec94fa1b471881c1a353d8ffa259bb4dd18cbd0b ./contracts/interfaces/spool/ISpoolExternal.sol
502d6c0e5604a2d69afff16500dad56aeebb86f5599b378bad85f04d8647cbf1 ./contracts/interfaces/spool/ISpoolDoHardWork.sol
88f865d071e6f89d5065b55b425d26c6a02b8e29563cd9b4b00945aeeaee2a97 ./contracts/interfaces/spool/ISpoolReallocation.sol
c91cf39434094aa788ce56b111605dee08c306dcf22d8f926e87c64ea0929765 ./contracts/interfaces/vault/IVaultRestricted.sol
898c70152f7c857c2925340bc1403c446ac9e937fe964f167d307ef56ec9e853 ./contracts/interfaces/vault/IVaultIndexActions.sol
4bbd8bcbe8c67d13acf7f3933796f8e702b6a829652ed1d3b456a9314350fb01 ./contracts/interfaces/vault/IVaultBase.sol
a218eb6b60bc2c30ea809faee79a02bf0227905d0ebc2718d40d34e7526354c3 ./contracts/interfaces/vault/IVaultImmutable.sol
e80189c2b0e03bfe789f46e026683c0e704d995546a7b7d4af5e7c4c48f064f6 ./contracts/interfaces/vault/IRewardDrip.sol
ecd708a9ea4fd4a6dc8b76e24845c5130b4cd6f26720fdeddcac6816ab1c6cd3 ./contracts/interfaces/vault/IVaultDetails.sol
82d3d395e7cac5a3c41714213c55da8443892f21a3541d4da46ae7ffc1206351 ./contracts/libraries/Hash.sol
9d3053818464f98b117d7124b15bbc0a9d408fbc39fa91806751bc48ba9cdcac ./contracts/libraries/Bitwise.sol
c07064ef207f5b910e9f96b95178626f1997b994eb64604402456a938d971b9f ./contracts/libraries/Math.sol
643bd7253bf132220842b8080b8e43f53e36863effb4aabe0775a46884430886 ./contracts/libraries/Max/128Bit.sol
```

Tests

```
85ed5151f59d85a5d3ff7f66172bd762787e9c4c20d1a1eb1d68a9a097ddef42 ./test/RiskProviderRegistery.spec.ts
f1fb6f72164e02d1e65a162d948bfe563c6c651f3a05b757fdfdbcc4e277d91e ./test/RewardDrip.spec.ts
d31cb0d18c0aa686922f45411754afde9b4ff25fbfeb2de01eb297eddc51ffa4 ./test/Vault.spec.ts
e3e60549551ba7ac6ebbf2031f315ae83fe91946cc1d02fc040df6a97707c0f9 ./test/Controller.spec.ts
bdaea9c1946ab7af58630f9b62e9a9d3fd4a9b2a9fadec1c5c71bf8446aa4227 ./test/FeeHandler.spec.ts
a531e1bf202338228f3a979949e6342d304bad488462f036182a5e8a994a3833 ./test/Spool.spec.ts
630837445e20609b610fa680a6ad061d33ba1dbd83e06b791fe0832d99db9ce7 ./test/Redistribution.spec.ts
2c02102e17474df2d1a7d412e17f77276cb34389bdcd275785a04754f4cf9130 ./test/FastWithdraw.spec.ts
8caaa9a24f48959818e6bedfc69cca4e653fe998377442baad3a3bc0d7b0d750 ./test/shared/fixtures.ts
e45a0c586bfb8e4418848fc7857a5092099e8291824c09684927b8e2eda30449 ./test/shared/constants.ts
579fe53c9ff2251f86f44417d3eb07849e72e3d04200e9be42fd9e9b5cf92c77 ./test/shared/utilities.ts
7e80c29ebe69ce871610cc136ca2222e967191ef6b88a1334cc31366db297035 ./test/shared/chaiExtension/chaiExtInterface.ts
```

```
5056921f7cbef6bedee5957fb0939487dd4cc94e7c1249c42a95c4dfd6860d41 ./test/shared/chaiExtension/chaiExtAssertions.ts
7e8468932be4b9fd7580fc272c2a619afed2d93c9036c617e9655809297a3648 ./test/strategies/unitTests/Aave.spec.ts
7e0b039f0ec96af53727cb50f859589813d2e43a7761f289ca94aa101da6aa03 ./test/strategies/unitTests/Harvest.spec.ts
d93d7ffdaf1d75b154cc3a1bdea033ad4be53c8d3b168b50e026200b9c3b2177 ./test/strategies/unitTests/Yearn.spec.ts
643a40d1fad990795fa80a50d4ae43171be21ef4b23320fe344991933669e819 ./test/strategies/unitTests/Idle.spec.ts
eddc3b005a3834f505cc32ce97d8365b8f4565305c5fe340ec9e9e9d8092a692 ./test/strategies/unitTests/CurveLG3pool.spec.ts
8129b8de7f816a30612d2e8aea3e7ae2297d0459c5d681f8541d2e335acdae80 ./test/strategies/unitTests/Masterchef.spec.ts
c3f8e57847da73f00605c287973572920aee9d4f952eac9a4caaa9c7f4b50dbb ./test/strategies/unitTests/Convex3pool.spec.ts
e6c28cbf88b4c9dcb101cb2b6e9486ae25c5733e17c003e219786772f9393a31 ./test/strategies/unitTests/Compound.spec.ts
909aeebe26847e136bba652bb369a070b11c11c239004a5ffa59a1ae1f44f8f6 ./test/strategies/unitTests/BarnBridgeMultiReward.spec.ts
5484ff9ac77c44fd329df2a0d9f0b9b38ffdb6b6e7f04ce9977fbb648a6fba0e ./test/strategies/unitTests/shared/stratSetupUtilities.ts
```

Changelog

- 2022-03-14 - Initial report (a9654cf)
- 2022-03-21 - Initial report update
- 2022-04-03 - Reaudit update (d6ec9e3)

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.