Quantstamp Security Assessment Certificate

October 18th 2021 — Quantstamp Verified

Solace

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

Executive Summary

Type	Investment Protocol
Auditors	Jan Gorzny, Blockchain Researcher Roman Rohleder, Research Engineer Hisham Galal, Research Engineer
Timeline	2021-09-27 through 2021-10-18
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Solace Docs
Documentation Quality	Undetermined
Test Quality	Undetermined

Source Code

Repository	Commit
core	928d61d

Total Issues	15 (11 Resolved)
High Risk Issues	2 (1 Resolved)
Medium Risk Issues	2 (2 Resolved)
Low Risk Issues	4 (2 Resolved)
Informational Risk Issues	3 (3 Resolved)
Undetermined Risk Issues	4 (3 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

Quantstamp has reviewed the Solace protocol and found several issues that were addressed. There are a few high- and medium-severity issues that may affect protocol behaviour, and some undetermined severity issues may also cause problems with user experience or protocol behaviour. Several other smaller issues, as well as best practices, would further increase confidence in the codebase's correctness. Almost all of the issues were all acknowledged or resolved.
Quantstamp was able to execute the tests for the project but two tests failed; Quantstamp very strongly recommends having a working test suite. The tests that failed may not be relevant to the files that were in scope for this audit. Quantstamp was not able to compute test coverage.
Disclaimer: Only some files were in scope for this audit, in particular, only the files whose hashes are included at the end of the report, were in scope.

ID	Description	Severity	Status
QSP-1	Missing Return Value Check	High	Acknowledged
QSP-2	Unsafe Cast Leading to Overflow	High	Fixed
QSP-3	Race Conditions / Front-Running	Medium	Fixed
QSP-4	Unlimited Allowance Anti-Pattern	Medium	Fixed
QSP-5	Missing Input Validation	Low	Unresolved
QSP-6	Unenforced Expected Bounds	Low	Fixed
QSP-7	Privileged Roles and Ownership	Low	Acknowledged
QSP-8	Use of Custom and Contradicting Library Implementations of `min`	Low	Fixed
QSP-9	Strict Inequality	Informational	Fixed
QSP-10	Events not Emitted on State Change	Informational	Fixed
QSP-11	Redundant/Wrong Statement Check Order	Informational	Fixed
QSP-12	Gas Usage / `for` Loop Concerns	Undetermined	Acknowledged
QSP-13	Potential Unexpected Revert	Undetermined	Fixed
QSP-14	Inconsistent Accounting of `_activeCoverAmount`	Undetermined	Fixed
QSP-15	Potential Gas Greifing	Undetermined	Mitigated

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 v0.6.6

Steps taken to run the tools:

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

Findings

QSP-1 Missing Return Value Check

Severity: *High Risk*

Status: Acknowledged

File(s) affected: `Vault.sol`, `Treasury.sol`, `ClaimsEscrow.sol`, `PolicyManager.sol`

Description: The following function calls make a call to a function that returns a value, which however is not checked:

1. `ClaimsEscrow.receiveClaim()` does not check the return value of `IVault(payable(_registry.vault())).requestEth()`.
2. `ClaimsEscrow.withdrawClaimsPayout()` does not check the return value of `IVault(payable(_registry.vault())).requestEth()`.
3. `Treasury._transferEth()` does not check the return value of `IVault(payable(_registry.vault())).requestEth()`.
4. `Treasury.swap()` does not check the return values of `token.approve()` or `_swapRouter.exactInput()`.
5. `PolicyManager.addProduct()` does not check the return value of `products.add(product);` and similarly, `PolicyManager.removeProduct()` does not check the return value of `products.remove(product)`.
6. `Vault.sol`: No return values checked for `safeTransferFrom` (e.g., line 93, 144). Other implementations (or future versions) of ERC20s may not revert if the transfer fails.

Recommendation: Check the return value (e.g., put the call in a `require` statement for boolean return values).

Update:

1. Fixed, by making `requestEth()` no longer return something
2. Fixed, by making `requestEth()` no longer return something
3. Fixed, by making `requestEth()` no longer return something
4. Fixed, by removing `Treasury.swap()` altogether
5. Acknowledged with "Return value from `EnumerableSet.add(item)` is of no significance to the success of `PolicyManager.addProduct(product)`."
6. Acknowledged with "OpenZeppelin's SafeERC20 performs the safety checks. Adding our own checks would be redundant; there are no return values even if we wanted to."

QSP-2 Unsafe Cast Leading to Overflow

Severity: *High Risk*

Status: Fixed

File(s) affected: `PolicyManager.sol`

Description: In `PolicyManager.updateActivePolicies()` in line 324 cover amount of a currently processed policy is cast to `int256` type, negated and then passed to `IProduct(product).updateActiveCoverAmount()`, which expects a `int256` type parameter: `IProduct(product).updateActiveCoverAmount(-int256(coverAmount));` Given that `coverAmount` is of `uint256` type and the primitive cast operation `int256()` is used, values for `coverAmount` greater than `type(int256).max` may lead to overflows.

Recommendation: Replace the primitive unsafe cast operation `int256(coverAmount)` in line 324 of `PolicyManager.sol` with for example its safe counter part `SafeCast.toInt256(coverAmount)` of OpenZeppelins SafeCast library.

Update: This has been resolved (by using `SafeCast`).

QSP-3 Race Conditions / Front-Running

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `Treasury.sol`

Description: A block is an ordered collection of transactions from all around the network. It's possible for the ordering of these transactions to manipulate the end result of a block. A miner attacker can take advantage of this by generating and moving transactions in a way that benefits themselves.
In particular, `Treasury.swap()` is vulnerable to price manipulation/front-running. In line 232 of `Treasury.sol`, the `swap()` function does not validate that parameter `amountOutMinimum` to be non-zero. Having `amountOutMinimum` being zero can be a significant risk in production, as an unusually bad price for a trade due to a front running sandwich or another type of price manipulation could impact the swap.

Recommendation: Consider adding a strict check on parameter `amountOutMinimum` on being non-zero, to prevent potentially arbitrarily bad swaps from happening.

Update: This has been resolved (by removing the function).

QSP-4 Unlimited Allowance Anti-Pattern

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `Treasury.sol`

Description: In line 240 of `Treasury.sol`, the `swap()` function calls approve with the maximum value `type(uint256).max`, which is not a good pattern for security.

Recommendation: Consider bounding the approval to be safe. And the allowance should be decreased whenever that allowance is not needed anymore.

Update: This has been resolved (by removing the function).

QSP-5 Missing Input Validation

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `Governable.sol`, `PolicyManager.sol`, `Registry.sol`, `SOLACE.sol`, `ClaimsEscrow.sol`, `RiskManager.sol`, `Treasury.sol`, `Vault.sol`, `BaseProduct.sol`

Description: `Governable.sol`:

- `constructor` does not check that `governance_` is not the zero address.
- `setGovernance` does not check that `newGovernance_` is not the zero address.

`PolicyManager.sol`:

- `createPolicy` does not check that `policyHolder` is not the zero address; it does not check if `coverAmount` and `price` are non-zero.
- `addProduct` does not check that `product` is not the zero address.
- `setPolicyDescriptor` does not check that `policyDescriptor_` is not the zero address.
- `setPolicyInfo` does not check if `coverAmount` and `price` are non-zero.

`Registry.sol`:

- `setWeth` does not check that `weth_` is not the zero address.
- `setVault` does not check that `vault_` is not the zero address.
- `setClaimsEscrow` does not check that `claimsEscrow_` is not the zero address.
- `setTreasury` does not check that `treasury_` is not the zero address.
- `setPolicyManager` does not check that `policyManager_` is not the zero address.
- `setRiskManager` does not check that `riskManager_` is not the zero address.
- `setSolace` does not check that `solace_` is not the zero address.
- `setMaster` does not check that `master_` is not the zero address.
- `setLocker` does not check that `locker_` is not the zero address.

`SOLACE.sol`:

- `addMinter` does not check that `minter` is not the zero address.

`ClaimsEscrow.sol`:

- `constructor` does not check that `registry_` is not the zero address.
- `receiveClaim` does not check that `claimant` is not the zero address.
- `listClaims` does not check that `claimant` is not the zero address.

`RiskManager.sol`:

- `constructor` does not validate that `registry_` is not the zero address.
- `assessRisk` does not validate that `prod` is not the zero address (in this case, a call to `productIsActive` on the parameter may also be sufficient).
- `addProduct` does not validate that `product_` is not the zero address.
- `setProductParams()` does not check that parameters `products_[i]` are not the zero address.
- `extendPolicy()` does not check that parameter `extension` is non-zero.
- `updatePolicy()` does not check that parameter `extension` is non-zero.

`Treasury.sol`:

- `constructor` does not validate that `registry_`, `governance_`, and `swapRouter_` are not the zero address.
- `spend` does not validate that `recipient` or `token` is not the zero address.
- `refund` does not validate that `user` is not the zero address.
- `_transferEth` does not validate that `user` is not the zero address.
- `swap` does not validate that `amountIn` or `amountOutMinimum` are not zero.

`Vault.sol`:

- `constructor` does not validate that `governance_` and `swapRouter_` are not the zero address.
- `setRequestor` does not validate that `dst` is not the zero address.

`BaseProduct.sol`:

- `constructor` does not validate that `governance_`, `policyManager_`, `registry_`, and `coveredPlatform_` are not the zero address.
- `buyPolicy` does not validate that `policyholder` is not the zero address.
- `addSigner` does not validate that `signer` is not the zero address.

- `setCoveredPlatform` does not validate that `coveredPlatform_` is not the zero address.
- `setPolicyManager` does not validate that `policyManager_` is not the zero address.

Recommendation: Check that the addresses are non-zero, make it clear that this corner case is acceptable, or have a plan for redeployment if the wrong address is set.

Update: This has been mostly resolved. The constructor in `Governable.sol` has not been changed. From the team:

- `Governable`: `setGovernance()` intentionally does not check zero address because it can be used to stop governance transfer. Added a zero address check to `acceptGovernance()`.
- `PolicyManager`: added policyholder zero address check to `BaseProduct.buyPolicy()`. Checks for zero `coverAmount` and price are performed in `BaseProduct.buyPolicy()`. Will not duplicate these checks in `PolicyManager`, in part because there are future use cases that may require zero cover amount and/or price. Will not add zero `coverAmount` and price checks to `setPolicyInfo` for same reasons. Will not check zero `policyDescriptor` as we may want to remove the descriptor. Also fixed natspec.
- `Registry`: added zero address checks on all setters.
- `Solace`: added zero address check on `addMinter()`
- `ClaimsEscrow`: Added zero address check on `registry` in constructor. Added zero address check on `claimant` in `receiveClaim()`. Added zero address check on `claimant` in `listClaims()`.
- `RiskManager`: Added zero address check on `registry` in constructor. Added zero address check on `product` in `addProduct()` and `setProductParams()`. Check for `productIsActive()` in `assessRisk()` is sufficient. `extendPolicy()` and `updatePolicy()` are `Product` functions.
- `Treasury`: Removed `swap()` and `swapRouter`. Added zero address check on `registry` in constructor. Added zero address check on `recipient` in `spend()` and `_transferEth()`. Added zero address check on `token` in `spend()`.
- `Vault`: Added zero address check on `registry` and `weth` in constructor. Added zero address check on `requestor` in `setRequestor()`.
- `BaseProduct`: Added zero address check for `registry`, `policyManager`, and `coveredPlatform` in constructor, `setPolicyManager()`, and `setCoveredPlatform()`. Added zero address check for `policyholder` in `buyPolicy()`. We allow for any extension in `extendPolicy()` and `updatePolicy()` as long as the new duration is within `minPeriod` and `maxPeriod`.

QSP-6 Unenforced Expected Bounds

Severity: *Low Risk*

Status: Fixed

File(s) affected: `BaseProduct.sol`

Description: In `constructor`, the values `minPeriod_` and `maxPeriod_` are set, but it is not required that the max period is at least as large as the min period. This is also not checked in functions `setMinPeriod` and `setMaxPeriod`.

Recommendation: Add a `require` statement that enforces the condition above in the appropriate places.

Update: This was resolved (by adding a check `minPeriod <= maxPeriod`).

QSP-7 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Vault.sol`, `BaseProduct.sol`, `Treasury.sol`, `RiskManager.sol`, `PolicyManager.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract.
Below we list some privileges in this project:
`PolicyManager.sol`: only the governance can add products; only a minter can mint tokens; only the governance can change claims and set cool down periods.
`RiskManager.sol`: there are special roles for adding and removing products.
`Treasury.sol`: there are special roles regarding refunds which are possible in some situations for products; governance can swap tokens.
`Vault.sol`: there is an emergency shutdown; governance can set some important addresses.
`BaseProduct.sol`: governance is in charge of adding signers, and pausing, and setting some important addresses.
As an example, we describe one of these roles in detail. The `governor`, as set by the `Governable.sol` contract, may perform the following privileged actions:

1. Appoint a new governor, by calling `Governable.setGovernance()` with the new address. However, the new governor must accept, by calling `Governable.acceptGovernance()`.
2. Set and change the contract addresses of contracts: WETH9, ClaimsEscrow, PolicyManager, RiskManager, SOLACE, Treasury, Vault, Locker and Master.
3. Change the maximum SOLACE supply to any arbitrary number, higher than the current supply.
4. Remove existing SOLACE minters or add arbitrary new ones.
5. Change the value of any claim to an arbitrary new value.
6. Transfer an arbitrary amount of ETH from the ClaimsEscrow contract to the Vault contract, or to any arbitrary address, by calling `Registry.setVault()` with the desired target address beforehand.
7. Change the cool down period between claim receival and claim payout withdrawal for all future withdrawals, including unlocking withdrawal immediately after receival and ‘infinite’ locking of payouts.
8. Unset or arbitrarily set the list of premium recipients with arbitrary weights for payouts, by calling `Treasury.setPremiumRecipients()`.
9. Transfer an arbitrary amount of ETH or any amount of ERC20 tokens from the Treasury contract to any arbitrary address, by calling `Treasury.spend()`.
10. Execute any custom code of another contract, that conforms to the function signature function `safeTransfer(address to, uint256 value)`, by calling `Treasury.spend()`.
11. Swap arbitrary tokens in arbitrary amounts using arbitrary pool-pairs over Uniswap using `Treasury.swap()` or wrap & unwrap ETH and WETH using `Treasury.wrap()` and `Treasury.unwrap()` respectively.
12. Set or unset the Emergency Shutdown mode of the Vault by calling `Vault.setEmergencyShutdown()`, where when in emergency shutdown mode, users may no longer deposit into the Vault and whitdrawals may be done outside of the cool down period.

13. Set arbitrary cool down windows, between 1 second and practical infinity, by calling `Vault.setCooldownWindow()`. Since only during cool down withdrawals may occur and only outside of cool down deposits may occur, setting the window to a very large value one may achieve a similar state to Emergency Shutdown without being actually in it.
14. Give or revoke arbitrary addresses the `Requestor` status, by calling `Vault.setRequestor()`. Such requestors may call `Vault.requestEth()` and thereby withdraw an arbitrary amount from the Vaults assets (its ETH and WETH).
15. Add or remove arbitrary addresses as Products in the PolicyManager contract by calling `PolicyManager.addProduct()` and `PolicyManager.removeProduct()`, respectively.
16. Add or remove arbitrary products with arbitrary price and weight values in the RiskManager contract, by calling `addProduct()` or `removeProduct()` respectively.
17. Change the `_partialReservesFactor` value via `RiskManager.setPartialReservesFactor()`, thereby impacting the minimum and maximum coverages.
18. Change the minimum and maximum block periods a product policy may be purchased for.
19. Add or remove authorized claim signer addresses for a product.
20. Set or unset a products pause-state, within which buying and extending policies are paused.
21. Change a products covered platform address, as well as a products associated policy manager contract address.

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: From the team:
We are believers in the transformative power of decentralization and acknowledge that the protocol in its current state provides more privilege to central roles than we would like. We will control these central roles by:

1. The governance role largely exists to set parameters in the short term post launch. As soon as we verify the safe operation of our protocol we will lock the governance role, implemented in commit 644a1945b61e86f38fc2b69c6d38021ed8261bf3.
2. The governance role will be controlled by either the core team multisig, the Solace Launch DAO, or the Solace Community DAO. No single user will be able to perform governance actions.

QSP-8 Use of Custom and Contradicting Library Implementations of `min`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Treasury.sol`, `Vault.sol`

Description: In order to prevent deviating behaviour or vulnerabilities, it is recommended to use well known and tested libraries for common utility computations. In line 281 of `Treasury.sol` and line 405 of `Vault.sol` a `min()` function is implemented, instead of re-using, for example, the `min()` implementation of OpenZeppelin's Math library. Further, the two implementations deviate from another:

- `Treasury.min()` returns `return a <= b ? a : b;`
- `Vault.min()` returns `return a < b ? a : b;`

Recommendation: Consider using for both cases the `min()` implementation of OpenZeppelin's Math library instead, whose internal implementation corresponds to that of `Vault.min()`.

Update: This has been resolved (by using `Math.min`).

QSP-9 Strict Inequality

Severity: *Informational*

Status: Fixed

File(s) affected: `Vault.sol`

Description: On line 300, the max cool down is required to be strictly larger than the minimum. However, it may be required or expected in the future that these values are in fact equal.

Recommendation: Consider using less than or equal to, rather than strictly less than.

Update: This has been resolved (by adding equality).

QSP-10 Events not Emitted on State Change

Severity: *Informational*

Status: Fixed

File(s) affected: `BaseProduct.sol`, `RiskManager.sol`, `PolicyManager.sol`, `Vault.sol`, `Treasury.sol`, `ClaimsEscrow.sol`, `Governable.sol`

Description: An event should always be emitted when a state change is performed in order to facilitate smart contract monitoring by other systems which want to integrate with the smart contract.
This is not the case for the functions:

1. `Governable.setGovernance()`
2. `ClaimsEscrow.adjustClaim()`
3. `ClaimsEscrow.returnEth()`
4. `ClaimsEscrow.setCooldownPeriod()`
5. `Treasury.routePremiums()`
6. `Treasury._transferEth()`
7. `Vault.startCooldown()`
8. `Vault.stopCooldown()`

9. `Vault.setCooldownWindow()`
10. `Vault.setRequestor()`
11. `PolicyManager.updateActivePolicies()`
12. `PolicyManager.setPolicyDescriptor()`
13. `RiskManager.setPartialReservesFactor()`
14. `BaseProduct.setMinPeriod()`
15. `BaseProduct.setMaxPeriod()`
16. `BaseProduct.setPaused()`
17. `BaseProduct.setCoveredPlatform()`
18. `BaseProduct.setPolicyManager()`

Recommendation: Emit an event in the aforementioned functions.

Update: This has been resolved. From the team:

- `Governable`: Renamed `setGovernance` to `setPendingGovernance()`, added event.
- `ClaimsEscrow`: Added `ClaimAdjusted`, `EthReturned`, and `CooldownPeriodSet` events.
- `Treasury`: Added `PremiumsRouted` and `EthRefunded` events.
- `Vault`: Added `CooldownStarted`, `CooldownStopped`, `CooldownWindowSet`, and `RequestorSet` events.
- `PolicyManager`: Did not add an event for `updateActivePolicies()` as each policy will be burned and emit `PolicyBurned` and Transfer to the zero address. Added `PolicyDescriptorSet` event
- `RiskManager`: Added `PartialReservesFactorSet` event.
- `BaseProduct`: Added `MinPeriodSet`, `MaxPeriodSet`, `PauseSet`, `CoveredPlatformSet`, and `PolicyManagerSet` events.

QSP-11 Redundant/Wrong Statement Check Order

Severity: *Informational*

Status: Fixed

File(s) affected: `Treasury.sol`

Description: In line 58 the `registry_` parameter of the `Treasury.sol` constructor is checked to be non-zero, before trying to get the Vault contract address: `if (registry_ != address(0) && _registry.vault() != address(0))`. As before in line 55 `registry_` is already used to retrieve the address for the WETH contract, `_weth = IWETH9(payable(_registry.weth()));`, in case of `registry_` being the zero address, it would have reverted anyway, making the check in line 58 unnecessary.

Recommendation: Consider removing the first part of the if statement from line 58, `if (registry_ != address(0)`, or moving it up into a separate check, i.e., line 53, before it's other use in line 55.

Update: This has been resolved.

QSP-12 Gas Usage / `for` Loop Concerns

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `PolicyManager.sol`, `ClaimsEscrow.sol`, `RiskManager.sol`, `Treasury.sol`

Description: Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely. It is best to break such loops into individual functions as possible.
`PolicyManager.sol`: `listPolicies` has a large loop.
`ClaimsEscrow.sol`: `listClaims` has a large loop.
`RiskManager.sol`: `setProductParams` has a for-loop.
`Treasury.sol`: `routePremiums`, `setPremiumRecipients` have a for-loop.
Of particular concern is the `setPremiumRecipients` function in `Treasury.sol`. In line 78 of `Treasury.sol` a for-loop is used to iterate over the length of the `_premiumRecipients[]` array. For sufficiently large arrays, processing over this array could run out of gas, potentially denying `Treasury.routePremiums()` the service of paying out to all recipients. Similarly, `Treasury.setPremiumRecipients()` iterates over it's parameter weights. Given the current logic of this function, it is not possible to split up larger arrays into multiple calls, also effectively preventing from using large weights (and therefore recipients) arrays.

Recommendation: Allow a method to iterate over a range of indices for each of the mentioned functions, or ensure that lists cannot get too large. Consider having a fixed maximum length for the recipients parameter in line 192 of `Treasury.sol`, after determining it using gas analysis for `Treasury.routePremiums()`.

Update: From the team:

- `PolicyManager.listPolicies()` and `ClaimsEscrow.listClaims()` are not intended to be executed on chain. You should use ERC721 Enumerable methods on chain.
- `RiskManager.setProductParams()` must be executed by governance. Governance is willing to pay the transaction fees.
- `Treasury.routePremiums()`: The plan is to have the Vault be the only recipient with the option to send the DAO a small protocol fee. Added a max length of 16 recipients in `setPremiumRecipients()`.

QSP-13 Potential Unexpected Revert

Severity: *Undetermined*

Status: Fixed

File(s) affected: `Vault.sol`

Description: On line 393, there is a comment that says the code may revert if a value is too large.

Recommendation: If this is problematic, ensure that this value cannot be so large, or ensure that the revert is handled safely and in a useful manner.

Update: This has been resolved (by removing the comment; "That comment was warning about tokens with excessively high supply. Since we use eth, that supply is unreachable").

QSP-14 Inconsistent Accounting of `_activeCoverAmount`

Severity: *Undetermined*

Status: Fixed

File(s) affected: `BaseProduct.sol`

Description: In `BaseProduct.sol` the currently covered amount (in wei) is tracked in `_activeCoverAmount` and increased or decreased when `BaseProduct.buyPolicy()` or `BaseProduct.cancelPolicy()` are called, respectively. However, when `BaseProduct.updateCoverAmount()` or when `BaseProduct.updatePolicy()` are called, which may increase or decrease the cover, the tracked `_activeCoverAmount` is not updated accordingly, leading to an inconsistent state between `_activeCoverAmount` and the actually currently active cover amount, as long as `PolicyManager.updateActivePolicies()` is not called explicitly afterwards.

Recommendation: Add corresponding updates to `_activeCoverAmount` in `BaseProduct.updateCoverAmount()` and `BaseProduct.updatePolicy()`.

Update: This has been resolved (by adding updates to the variables).

QSP-15 Potential Gas Greifing

Severity: *Undetermined*

Status: Mitigated

File(s) affected: `Treasury.sol`

Description: The loop on line 83 of `Treasury.sol` may be subject to gas griefing, that is, if one of the accounts that the `call` is made on has a fallback which uses all the gas, the call will fail (and subsequent accounts will not get their Eth).

Recommendation: Allow indices to be specified, specify a gas limit per `call`, or ensure that accounts are not contracts. Alternatively, use the pull pattern (instead of push).

Update: This has been mitigated (by limiting the gas per call).

Code Documentation

1. In line 368 of `Vault.sol` the comment for `Vault._totalAssets()` states that it is also including assets loaned out to Strategies. However, this is not represented in the code, rather it returns the sum of the Vault's current WETH and ETH balance. **Update:** resolved.
2. In line 323 of `Vault.sol` the comment for the return parameter speaks of a variable `tokens`, but the actually returned variable is `shares`. **Update:** Instead of renaming tokens to shares in the heading of the comment of `Vault._withdraw()`, the team should have renamed tokens to shares for the return parameter comment in `Vault._deposit()`, to be in line with the variable naming of the returned variable.
3. In line 238 of `PolicyManager.sol` the parameters in the `PolicyManager.createPolicy()` functions' NatSpec comment are in the order `policyholder`, `expirationBlock`, `coverAmount`, `price`, `positionDescription`. However, the actual function parameters are in the order `policyholder`, `coverAmount`, `expirationBlock`, `price`, `positionDescription`. The order should be fixed, as also the generated online documentation contains the wrong order and may lead to wrong usage. Similarly the same issue is present in line 270 of `PolicyManager.sol` for function `setPolicyInfo()`. **Update:** resolved.
4. In line 359 of `BaseProduct.sol` the comment for function `paused()` states "Cannot buy new policies while paused. (Default is False)", which seems to be a copy-and-paste from line 44 and should be, to be more precise, for example "Returns whether or not product is currently in paused state." **Update:** resolved.

Adherence to Best Practices

1. As a general suggestion, in require error messages, adding messages of the form "contract:function" makes it easier to spot which function reverted the transaction.
2. `SOLACE.sol`: Constant on line 31 has too many zeros. Use underscores or exponents. **Update:** resolved.
3. `RiskManager.sol`: On lines 88 and 176, there is a magic constant. **Update:** resolved.
4. `RiskManager.sol`: function `setProductParams` duplicates much of the work of `addProduct`; consider refactoring the code to include a helper function for easier maintenance. **Update:** The team has said "[they] would create the helper function but the current setup uses less gas."
5. `Treasury.sol`: On line 236, there is a magic constant (in hex). **Update:** resolved.
6. `Vault.sol`: On lines 159 and 160, constants have too many zeros. Consider exponentiation or the use of underscores. Alternatively, for readability the lines should be changed to `1 ether;`. **Update:** resolved.
7. `Vault.sol`: Line 45 sets `uint40 internal _cooldownMin = 604800;`. For readability the line should be changed to `uint40 internal _cooldownMin = 7 days;`. **Update:** resolved.
8. `Vault.sol`: Line 48 sets `uint40 internal _cooldownMax = 3024000;`. For readability the line should be changed to `uint40 internal _cooldownMax = 35 days;`. **Update:** resolved.
9. `ClaimsEscrow.sol`: line 33 sets `uint256 internal _cooldownPeriod = 3600; // one hour.` For readability the line could be changed to `uint256 internal _cooldownPeriod = 1 hours;`. **Update:** resolved.
10. `BaseProduct.sol`: On lines 125, 163, 165, 198, 238, 240, 266, 323 there is a magic constant. **Update:** resolved.
11. For easier verifiability and user experience it is recommended to have sensitive address state variables with a public visibility modifier. Therefore the following addresses should be changed from private/internal visibility to public visibility:

    - `Governable.sol`: `_governance`, `_newGovernance`.

. `Registry.sol`: `_weth`, `_vault`, `_claimsEscrow`, `_treasury`, `_policyManager`, `_riskManager`, `_solace`, `_master`, `_locker`.

. `ClaimsEscrow.sol`: `_registry`.

. `Treasury.sol`: `_registry`, `_swapRouter`, `_weth`.

. `Vault.sol`: `_weth`, `_registry`.

. `PolicyManager.sol`: `_policyDescriptor`, `products`.

. `RiskManager.sol`: `_registry`.

. `BaseProduct.sol`: `_policyManager`, `_registry`, `_coveredPlatform`.

. **Update:**: From the team: "Those variables are explicitly available as external view functions. It also takes less gas this way".

12. For better disambiguation the error message in modifier `onlyNewGovernance` should be changed from `!governance` to for example `!newGovernance`, as it checks for `_newGovernance`, differentiating it from modifier `onlyGovernance`, which already has this error message. **Update:** resolved.

13. It is recommended to remove unused variables. As state variable `ETH_ADDRESS` in line 51 of `ClaimsEscrow.sol` is unused, it should be removed. **Update:** resolved.

14. In line 200 of `RiskManager.sol` the function `addProduct()` not only adds new products, but is also able to change the parameters of an existing product. For better disambiguation the function should be renamed to reflect this fact to, for example, `addOrChangeProduct()`. **Update:** the team has acknowledged this suggestion: "[We] agree that your suggestion would be less ambiguous, but the current setup is more aligned with our ops."

15. In line 274 of `RiskManager.sol` the function `setProductParams()` does not set/change the parameters of existing products (`RiskManager.addProduct()` can be used for that), but rather first purges all existing products and then adds the new list of products that are provided as parameters. For better disambiguation the function should be renamed to better represent this fact to, for example, `purgeAndAddProducts()` or the two functionalities should be separated into `purgeProducts()` and `addProducts()`. **Update:** the team has acknowledged this suggestion: "[We] agree that your suggestion would be less ambiguous, but the current setup is more aligned with our ops."

16. For improved readability it is recommended to have a maximum line length of 79 or 99. Therefore following lines should be shortened accordingly, which exceed these limits:

    . `Governable.sol`: lines 9, 11, 12 and 13.

    . `Registry.sol`: lines 12 and 14.

    . `SOLACE.sol`: lines 13, 14 and 30.

    . `ClaimsEscrow.sol`: lines 20, 22, 24, 32, 58, 68, 74, 97, 100, 102, 136, 146, 166, 174, 185, 203, 218, 247 and 257.

    . `Treasury.sol`: lines 19, 21, 23, 52, 130, 158, 192, 216 and 232.

    . `Vault.sol`: lines 19, 21, 23, 61, 71, 84, 114, 132, 164, 245, 294, 299, 306, 352, 368, 388, 410 and 412.

    . `PolicyManager.sol`: lines 17, 19, 31, 67, 82, 84, 92, 101, 110, 119, 128, 138, 148, 218, 409 and 419.

    . `RiskManager.sol`: lines 15, 17, 19, 21, 58, 60, 65, 151, 153, 176, 198, 200, 269, 272 and 313.

    . `BaseProduct.sol`: lines 19, 21, 42, 78, 82, 118, 122, 131, 144, 149, 151, 159, 179, 190, 192, 209, 215, 221, 223, 228, 236, 240, 261, 275, 277, 293, 299, 307, 321, 322, 342 and 386.

    . **Update:** the team has acknowledged this suggestion: "[We] agree that short lines are easier to read but not enough to be worth the effort, ignoring."

Test Results

Test Suite Results

Although Quantstamp was able to execute tests, two tests failed. These may not be relevant to the files that were in scope for this audit; similarly, the test suite tests code which was not in the scope of this audit.

```
AaveV2ProductKovan
  covered platform
     ✓ starts as aave data provider
     ✓ cannot be set by non governor
     ✓ can be set
  position description
     ✓ cannot be zero length
     ✓ cannot be odd size
     ✓ cannot have non aTokens
     ✓ can be one or more aTokens
  implementedFunctions
     ✓ can getQuote
     ✓ cannot buy policy with invalid description
     ✓ can buyPolicy
     ✓ can buy duplicate policy
     ✓ can buy policy that covers multiple positions
     ✓ can get product name
  submitClaim
     ✓ cannot submit claim with expired signature
     ✓ cannot submit claim on someone elses policy
     ✓ cannot submit claim on someone elses policy after transfer
     ✓ cannot submit claim signed for someone else
     ✓ cannot submit claim from wrong product
     ✓ cannot submit claim with excessive payout
     ✓ cannot submit claim with forged signature
     ✓ cannot submit claim from unauthorized signer
     ✓ cannot submit claim with changed arguments
     ✓ cannot submit claim with invalid domain
     ✓ cannot submit claim with invalid typehash
     ✓ can open a claim
       ✓ aAAVE
       ✓ aBAT
       ✓ aBUSD
       ✓ aDAI
       ✓ aENJ
       ✓ aKNC
       ✓ aLINK
```

```
        ✓ aMANA
        ✓ aMKR
        ✓ aREN
        ✓ aSNX
        ✓ aSUSD
        ✓ aTUSD
        ✓ aUSDC
        ✓ aUSDT
        ✓ aWBTC
        ✓ aWETH
        ✓ aYFI
        ✓ aZRX
        ✓ aUNI
         Error: Transaction reverted without a reason string
         at <UnrecognizedContract>.<unknown> (0x075a36ba8846c6b6f53644fdd3bf17e5151789dc)
         at <UnrecognizedContract>.<unknown> (0x4ccf2b08d747ab0bba99437f876f7e14e46749de)
         at <UnrecognizedContract>.<unknown> (0xe0fba4fc209b4948668006b2be61711b7f465bae)
         at <UnrecognizedContract>.<unknown> (0xe0fba4fc209b4948668006b2be61711b7f465bae)
         at async HardhatNode._mineBlockWithPendingTxs (/Users/jangorzny/Downloads/core-928d61dbb7337c607e7a30e3f404e67a6dc6eb54/node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
         at async HardhatNode.mineBlock (/Users/jangorzny/Downloads/core-928d61dbb7337c607e7a30e3f404e67a6dc6eb54/node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:435:16)
         at async EthModule._sendTransactionAndReturnHash (/Users/jangorzny/Downloads/core-928d61dbb7337c607e7a30e3f404e67a6dc6eb54/node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1494:18)
         at async HardhatNetworkProvider.request (/Users/jangorzny/Downloads/core-928d61dbb7337c607e7a30e3f404e67a6dc6eb54/node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:108:18)
         at async EthersProviderWrapper.send (/Users/jangorzny/Downloads/core-928d61dbb7337c607e7a30e3f404e67a6dc6eb54/node_modules/@nomiclabs/hardhat-ethers/src/internal/ethers-provider-wrapper.ts:13:20)

        ✓ aAMPL
         AssertionError: Expected "0" to be equal 1000000000
         at Context.<anonymous> (/Users/jangorzny/Downloads/core-928d61dbb7337c607e7a30e3f404e67a6dc6eb54/test/AaveV2ProductKovan.test.ts:386:70)
         at runMicrotasks (<anonymous>)
         at processTicksAndRejections (internal/process/task_queues.js:97:5)
         at runNextTicks (internal/process/task_queues.js:66:3)
         at listOnTimeout (internal/timers.js:523:9)
         at processTimers (internal/timers.js:497:7)

supported atokens:
  - aAAVE
  - aBAT
  - aBUSD
  - aDAI
  - aENJ
  - aKNC
  - aLINK
  - aMANA
  - aMKR
  - aREN
  - aSNX
  - aSUSD
  - aTUSD
  - aUSDC
  - aUSDT
  - aWBTC
  - aWETH
  - aYFI
  - aZRX

unsupported atokens:
  - aUNI
  - aAMPL


     1) should support all atokens

  BaseProduct
    deployment
       ✓ reverts zero addresses
       ✓ reverts invalid period
    governance
       ✓ starts with the correct governor
       ✓ rejects setting new governance by non governor
       ✓ can set new governance
       ✓ rejects governance transfer by non governor
       ✓ can transfer governance
  productParameters
     ✓ can get minPeriod
     ✓ can set minPeriod
     ✓ should revert setMinPeriod if not called by governance
     ✓ should revert setMinPeriod if greater than maxPeriod
     ✓ can get maxPeriod
     ✓ can set maxPeriod
     ✓ should revert setMaxPeriod if not called by governance
     ✓ should revert setMaxPeriod if greater than maxPeriod
     ✓ can get covered platform
     ✓ can set covered platform
     ✓ should revert setCoveredPlatform if not called by governance
     ✓ should revert setCoveredPlatform to zero address
     ✓ can get policy manager
     ✓ can set policy manager
     ✓ should revert setPolicyManager if not called by governance
     ✓ should revert setPolicyManager if not called by governance
  pause
     ✓ starts unpaused
     ✓ cannot be paused by non governance
     ✓ can be paused
     ✓ cannot be unpaused by non governance
     ✓ can be unpaused
  buyPolicy
     ✓ can getQuote
     ✓ cannot buy policy for zero address
     ✓ cannot buy policy with zero cover value
     ✓ cannot buy policy over max cover amount per product
     ✓ cannot buy policy over max cover amount per policy
     ✓ cannot buy policy with insufficient payment
     ✓ cannot buy policy under min period
     ✓ cannot buy policy over max period
     ✓ cannot buy policy while paused
     ✓ can buyPolicy
     ✓ returns overpayment from buy policy
  extendPolicy
     ✓ cannot extend nonexistent policy
     ✓ cannot extend someone elses policy
     ✓ cannot extend someone elses policy after transfer
     ✓ cannot extend from a different product
     ✓ cannot extend an expired policy
     ✓ cannot over extend policy
     ✓ cannot extend policy with insufficient payment
     ✓ cannot extend policy while paused
     ✓ can extend policy
     ✓ returns overpayment from extend policy
     ✓ can extend your policy after transfer
  updateCoverAmount
     ✓ cannot update cover amount while paused
     ✓ cannot update cover amount for nonexistent policy
     ✓ cannot update cover amount for someone elses policy
     ✓ cannot update cover amount for someone elses policy after transfer
     ✓ cannot update cover amount for from a different product
     ✓ cannot update cover amount for an expired policy
     ✓ cannot update cover amount to zero
     ✓ cannot update cover amount over max global cover amount
     ✓ cannot update cover amount over max user cover amount
     ✓ reverts insufficient payment
     ✓ can increase cover amount with exact payment
     ✓ can increase cover amount and return over payment
     ✓ can decrease cover amount
     ✓ can decrease cover amount and return msg.value
     ✓ can keep cover amount the same
     ✓ can update cover amount after transfer
  updatePolicy
     ✓ cannot update while paused
     ✓ cannot update nonexistent policy
     ✓ cannot update someone elses policy
     ✓ cannot update someone elses policy after transfer
```

```
      ✓ cannot update from a different product
      ✓ cannot update an expired policy
      ✓ cannot over extend policy
      ✓ cannot update policy with insufficient payment
      ✓ cannot update policy to zero cover amount
      ✓ cannot update over max global cover amount
      ✓ cannot update over max user cover amount
      ✓ can increase cover amount and extend
      ✓ returns overpayment from update policy
      ✓ can decrease cover amount
      ✓ can decrease cover amount and return msg.value
      ✓ can keep cover amount the same
      ✓ can update policy after transfer
    cancelPolicy
      ✓ cannot cancel nonexistent policy
      ✓ cannot cancel someone elses policy
      ✓ cannot cancel someone elses policy after transfer
      ✓ cannot cancel from a different product
      ✓ can cancel and refunds proper amount
      ✓ can cancel policy after transfer
    paclas signers
      ✓ non governance cannot add signers
      ✓ cannot add zero signer
      ✓ can add signers
      ✓ non governance cannot remove signers
      ✓ can remove signers
    active cover amount
      ✓ starts at zero
      ✓ cannot update by non policy manager
      ✓ can update
      ✓ cannot be negative

  ClaimsEscrow
    deployment
      ✓ should set the governance address
      ✓ should revert if registry is zero address
    setPendingGovernance
      ✓ should allow governance to set new governance address
      ✓ should revert if not called by governance
    receiveClaim
      ✓ should revert if not called by the vault
      ✓ should revert if zero claimant
      ✓ should create a Claim object with the right data
    withdrawClaimsPayout
      ✓ should revert if invalid claimID
      ✓ should revert if not called by the claimant
      ✓ should revert if cooldown period has not elapsed
      ✓ should transfer claim amount to claimant
      ✓ should emit ClaimWithdrawn event after function logic is successful
      ✓ should delete the Claim object after successful withdrawal
      ✓ should request more eth if needed
    adjustClaim
      ✓ should revert if not called by governance
      ✓ should revert if claim doesnt exist
      ✓ should update claim object with the right data
    returnEth
      ✓ should revert if not called by governance
      ✓ should returnEth
    cooldown
      ✓ should start at one hour
      ✓ should revert setCooldown if called by non governance
      ✓ should set cooldown
    isWithdrawable
      ✓ non existant claim should not be withdrawable
      ✓ new claim should not be withdrawable
      ✓ claim should become withdrawable
    timeLeft
      ✓ reverts non existant claim
      ✓ need to wait entire cooldown period for a new claim
      ✓ counts down
      ✓ hits zero
    listClaims
      ✓ lists claims
      ✓ does not list claims for zero address
    transfer
      ✓ should reject transfer of nonexistent token
      ✓ should reject transfer by non owner
      ✓ should transfer
      ✓ should reject safeTransfer of nonexistent token
      ✓ should reject safeTransfer by non owner
      ✓ should safeTransfer

  CpFarm
    farm creation
      ✓ can create farms
      ✓ returns farm information
    governance
      ✓ starts with the correct governor
      ✓ rejects setting new governance by non governor
      ✓ can set new governance
      ✓ rejects governance transfer by non governor
      ✓ can transfer governance
    deposit and withdraw
      ✓ can deposit eth
      ✓ can deposit eth via receive
      ✓ can deposit eth via fallback
      ✓ cannot deposit eth when lacking funds
      ✓ can deposit cp
      ✓ can deposit cp via permit
      ✓ cannot deposit cp when lacking funds
      ✓ cannot withdraw another user's rewards
      ✓ can withdraw rewards
      ✓ can withdraw cp
      ✓ cannot overwithdraw
    updates
      ✓ can update a single farm
      ✓ can set end
    rewards
      ✓ provides rewards to only farmer
      ✓ fairly provides rewards to all farmers
      ✓ does not distribute rewards before farm start
      ✓ does not distribute rewards after farm end
    safe rewards
      ✓ tracks unpaid rewards
      ✓ pays when funds are available
    edge cases
      ✓ can receive eth from vault via receive()
      ✓ can receive eth from vault via fallback()
      ✓ rejects setRewards by non master
      ✓ can get multiplier
    compound rewards
      ✓ does nothing if no rewards to compound
      ✓ users can compound rewards

  Governance
    governance
      ✓ starts with the correct governor
      ✓ rejects setting new governance by non governor
      ✓ can set new governance
      ✓ rejects governance transfer by non governor
      ✓ can transfer governance
      ✓ rejects transferring governance to the zero address

  LpAppraisor
    governance
      ✓ starts with the correct governor
      ✓ rejects setting new governance by non governor
      ✓ can set new governance
      ✓ rejects governance transfer by non governor
      ✓ can transfer governance
```

```
  appraisal
    ✓ cannot appraise non existant tokens
    ✓ appraises tokens
  incentive tuning
    ✓ non governance cannot tune the curve
    ✓ governance can tune the incentive curve
    ✓ non governance cannot replace the appraisal contract
    ✓ governance can replace the appraisal contract

Master
  governance
    ✓ starts with the correct governor
    ✓ rejects setting new governance by non governor
    ✓ can set new governance
    ✓ rejects governance transfer by non governor
    ✓ can transfer governance
  farm creation
    ✓ can create cp farms
    ✓ can create uniswap farms
    ✓ rejects farm creation by non governor
    ✓ rejects duplicate farm registration
    ✓ returns farm information
  updates
    ✓ can mass update
  rewards
    ✓ creates multiple farms
    ✓ fairly provides rewards to all farmers on all farms
    ✓ can change allocation points of farms
    ✓ can change solace per block
    ✓ can extend farms
    ✓ ends farms properly
    ✓ allows farmers to cash out
    ✓ allows farmers to withdraw rewards from multiple farms

PolicyManager
  ✓ has a correct name
  ✓ has a correct symbol
  ✓ has no policies
  ✓ has no nft token descriptor
  governance
    ✓ starts with the correct governor
    ✓ rejects setting new governance by non governor
    ✓ can set new governance
    ✓ rejects governance transfer by non governor
    ✓ can transfer governance
    ✓ rejects setting new nft token descriptor by non governor
    ✓ can set new nft token descriptor
  products
    ✓ starts with no products
    ✓ cannot add zero address
    ✓ can add products
    ✓ returns products
    ✓ rejects adds and removes by non governor
    ✓ can remove products
  policies
    ✓ non product cannot create policy
    ✓ can create policy

    2) can get policy info
    ✓ cannot update nonexistent policy
    ✓ product cannot update other products policy
    ✓ can set policy info
    ✓ can list my policies
    ✓ cannot directly burn policy
    ✓ can burn policy via product
    ✓ policy holder is token owner
  lifecycle
    ✓ pre-mint
    ✓ pre-expiration
    ✓ post-expiration
    ✓ post-burn
  updateActivePolicies
    ✓ can update active policies
  transfer
    ✓ should reject transfer of nonexistent token
    ✓ should reject transfer by non owner
    ✓ should transfer
    ✓ should reject safeTransfer of nonexistent token
    ✓ should reject safeTransfer by non owner
    ✓ should safeTransfer
  tokenURI
    ✓ can get tokenURI
    ✓ cannot get tokenURI for nonexistant policy id

Registry
  governance
    ✓ starts with the correct governor
    ✓ rejects setting new governance by non governor
    ✓ can set new governance
    ✓ rejects governance transfer by non governor
    ✓ can transfer governance
  weth
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  vault
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  claimsEscrow
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  treasury
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  policyManager
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  riskManager
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  solace
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  master
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address
  locker
    ✓ starts as the zero address
    ✓ can be set
    ✓ cannot be set by non governor
    ✓ cannot be set to the zero address

RiskManager
  deployment
    ✓ should revert if registry is zero address
```

```
  governance
    ✓ starts with the correct governor
    ✓ rejects setting new governance by non governor
    ✓ can set new governance
    ✓ rejects governance transfer by non governor
    ✓ can transfer governance
  product risk parameters
    ✓ should start unset
    ✓ should reject change by non governor
    ✓ should reject invalid inputs
    ✓ should be set with setProductParams()
    ✓ should be set with addProduct()
    ✓ should delete old products with setProductParams()
    ✓ should change weight with addProduct()
    ✓ should remove products
  max cover amount
    ✓ no assets no cover
    ✓ can cover
  partialReservesFactor
    ✓ starts at 10000 bps
    ✓ cannot be set by non governance
    ✓ can be set
  minCapitalRequirement
    ✓ should start at zero
    ✓ should track policy cover amount
    ✓ should leverage
  assess risk
    ✓ cannot accept risk from unregistered products
    ✓ can accept risk at max cover per product
    ✓ cannot accept risk over max cover per product
    ✓ can accept risk at max cover per policy
    ✓ cannot accept risk over max cover per policy
  sellable cover per product
    ✓ should revert on non products
    ✓ should be zero for inactive products
    ✓ should return correct amount

SOLACE
  deployment
    ✓ has a correct name
    ✓ has a correct symbol
    ✓ has 18 decimals
    ✓ has a correct governance
  _mint
    ✓ rejects a null account
    for a non zero account
      ✓ increments totalSupply
      ✓ increments recipient balance
      ✓ emits Transfer event
  mint
    ✓ allows minters to mint
    ✓ reverts mint() called by non-minters
    ✓ has a soft cap
    ✓ can mint up to the cap
    ✓ cannot mint more than the cap
    ✓ non governance cannot change cap
    ✓ governance can change cap
    ✓ can mint to new cap
    ✓ cannot lower cap under current supply
  minters
    ✓ governor is minter
    ✓ can add minters
    ✓ can remove minters
    ✓ reverts when !governance adds / removes minters
    ✓ cannot add zero address minter
  governance
    ✓ can transfer governance
    ✓ reverts governance transfers by non-governor

SolaceEthLpFarm
  farm creation
    ✓ can create farms
    ✓ returns farm information
  governance
    ✓ starts with the correct governor
    ✓ rejects setting new governance by non governor
    ✓ can set new governance
    ✓ rejects governance transfer by non governor
    ✓ can transfer governance
  deposit and withdraw
    ✓ can deposit
    ✓ can deposit via permit
    ✓ can deposit via mintAndDeposit
    ✓ cannot deposit when lacking funds
    ✓ cannot deposit tokens from other pools
    ✓ cannot withdraw another user's rewards
    ✓ can withdraw rewards
    ✓ can withdraw deposited tokens
    ✓ cannot overwithdraw
  updates
    ✓ can update a single farm
    ✓ can set end
  rewards
    ✓ accounts for token values
    ✓ only values tokens in range
    ✓ may lag behind pool swap left
    ✓ catches up to pool swap left after update
    ✓ may lag behind pool swap right
    ✓ catches up to pool swap right after update
    ✓ can withdraw rewards
    ✓ can withdraw stake
  beginning and end
    ✓ does not distribute rewards before farm start
    ✓ does not distribute rewards after farm end
  safe rewards
    ✓ tracks unpaid rewards
    ✓ pays when funds are available

Treasury
  deployment
    ✓ reverts if zero registry
    ✓ reverts if zero weth
    ✓ routes to vault if set
    ✓ routes to treasury if vault not set
  governance
    ✓ starts with the correct governor
    ✓ rejects setting new governance by non governor
    ✓ can set new governance
    ✓ rejects governance transfer by non governor
    ✓ can transfer governance
  deposit
    ✓ can deposit solace
    ✓ can deposit eth via receive
    ✓ can deposit eth via fallback
    ✓ can deposit weth
    ✓ can deposit other token
  wrap
    ✓ non governor cannot wrap eth
    ✓ can wrap eth
    ✓ non governor cannot unwrap eth
    ✓ can unwrap eth
  spend
    ✓ non governor cannot spend
    ✓ cannot spend zero address token
    ✓ cannot spend to zero address recipient
    ✓ can spend solace
    ✓ can spend unswapped token
    ✓ can spend eth
  route premiums
    ✓ with no recipients
```

```
      ✓ can route premiums with no recipients
      ✓ non governor cannot set recipients
      ✓ validates recipients and weights
      ✓ can set recipients
      ✓ can route premiums
      ✓ is safe from gas griefing
    refund
      ✓ non product cannot refund
      ✓ cannot refund to zero address
      ✓ product can refund in full
      ✓ product can partially refund
    treasury with vault as a premium recipient
      ✓ vault is a premium recipient
      ✓ can route premiums to vault
      ✓ can refund from vault
    weth
      ✓ can mint via receive
      ✓ can mint via fallback
      ✓ can mint via deposit
      ✓ can withdraw

  Vault
    deployment
      ✓ should set the right token name and symbol
      ✓ should set the governance address
      ✓ should initialize DOMAIN_SEPARATOR correctly
      ✓ reverts if zero registry
      ✓ reverts if zero weth
    setPendingGovernance
      ✓ should allow governance to set new governance address
      ✓ should revert if not called by governance
    setEmergencyShutdown
      ✓ should revert if not called by governance
      ✓ should successfully toggle emergency shutdown state in Vault
    setCooldownWindow
      ✓ should return correct initial values
      ✓ should revert if not set by governance
      ✓ should revert if invalid window
      ✓ should successfully set cooldown window
    pricePerShare
      ✓ should initially return 1:1 ETH-SCP
      ✓ should stay at 1:1 on first deposit
      ✓ should appreciate on sale of coverage
      ✓ should stay the same on deposit
      ✓ should stay the same on withdraw
      ✓ should depreciate on payout of claims
      ✓ should stay the same on deposit
      ✓ should stay the same on withdraw
    maxRedeemableShares
      ✓ should initially return zero
      ✓ should return the correct maxRedeemableShares - user can withdraw entire CP token balance
      ✓ should return the correct maxRedeemableShares - user can withdraw up to a portion of their CP token balance
    deposit eth
      ✓ revert if vault is in emergency shutdown
      ✓ should mint the first depositor CP tokens with ratio 1:1
      ✓ should mint WETH to the Vault
      ✓ should mint the second depositor CP tokens according to existing pool amount
      ✓ should emit Transfer event as CP tokens are minted
      ✓ should emit DepositMade event after function logic is successful
      ✓ should restart cooldown
      ✓ should not mint on receive()
      ✓ should not mint on fallback()
    deposit weth
      ✓ revert if vault is in emergency shutdown
      ✓ should mint the first depositor CP tokens with ratio 1:1
      ✓ should mint WETH to the Vault
      ✓ should mint the second depositor CP tokens according to existing pool amount
      ✓ should emit Transfer event as CP tokens are minted
      ✓ should emit DepositMade event after function logic is successful
      ✓ should restart cooldown
    transfer
      ✓ can transfer between non cooldown accounts
      ✓ should revert if accounts are in cooldown
      ✓ does not care about cooldown while in emergency shutdown
    withdraw eth
      ✓ should revert if withdrawer tries to redeem more shares than they own
      ✓ should revert if withdrawal brings Vault's totalAssets below the minimum capital requirement
      ✓ should revert if cooldown not started
      ✓ should revert if not enough time passed
      ✓ should revert if too much time passed
      ✓ should withdraw if correct time passed
      ✓ should unwrap weth if necessary
      while vault is in emergency shutdown
        ✓ does not care about mcr
        ✓ does not care about cooldown period
    withdraw weth
      ✓ should revert if withdrawer tries to redeem more shares than they own
      ✓ should revert if withdrawal brings Vault's totalAssets below the minimum capital requirement
      ✓ should revert if cooldown not started
      ✓ should revert if not enough time passed
      ✓ should revert if too much time passed
      ✓ should withdraw if correct time passed
      ✓ should wrap eth if necessary
      while vault is in emergency shutdown
        ✓ does not care about mcr
        ✓ does not care about cooldown period
    requestors
      ✓ should start with no authorized requestors
      ✓ should revert add and remove requestors by non governance
      ✓ should add and remove requestors
      ✓ cannot add zero address requestor
    requestEth
      ✓ should revert if not called by a requestor
      ✓ should send eth
      ✓ should get available eth
      ✓ can get zero eth
    share value
      ✓ deposits and withdraws at same value from start
      ✓ deposits and withdraws at same value from state n with gains
      ✓ deposits and withdraws at same value from state n with losses
      ✓ cannot get flashloan attacked

  WaaveProduct
    covered platform
      ✓ starts as waRegistry
      ✓ cannot be set by non governor
      ✓ can be set
    position description
      ✓ cannot be zero length
      ✓ cannot be odd size
      ✓ cannot have non waTokens
      ✓ can be one or more waTokens
    implementedFunctions
      ✓ can getQuote
      ✓ cannot buy policy with invalid description
      ✓ can buyPolicy
      ✓ can buy duplicate policy
      ✓ can buy policy that covers multiple positions
      ✓ can get product name
    submitClaim
      ✓ cannot submit claim with expired signature
      ✓ cannot submit claim on someone elses policy
      ✓ cannot submit claim on someone elses policy after transfer
      ✓ cannot submit claim signed for someone else
      ✓ cannot submit claim from wrong product
      ✓ cannot submit claim with excessive payout
      ✓ cannot submit claim with forged signature
      ✓ cannot submit claim from unauthorized signer
      ✓ cannot submit claim with changed arguments
      ✓ cannot submit claim with invalid domain
```

```
      ✓ cannot submit claim with invalid typehash
      ✓ can open a claim on a waWETH position
        ✓ waWETH
        ✓ waDAI
        ✓ waUSDT
        ✓ waWBTC
      ✓ should support all watokens
```

Solc version: 0.8.6	Optimizer enabled: true	Runs: 800	Block limit: 30000000 gas

Methods						
Contract	Method	Min	Max	Avg	# calls	usd (avg)
AaveV2Product	addSigner	-	-	47322	1	-
AaveV2Product	buyPolicy	360937	441965	370749	27	-
AaveV2Product	setCoveredPlatform	-	-	35244	2	-
AaveV2Product	submitClaim	281612	291453	282121	60	-
ClaimsEscrow	acceptGovernance	-	-	28185	2	-
ClaimsEscrow	adjustClaim	-	-	38774	2	-
ClaimsEscrow	receiveClaim	182397	249392	225474	32	-
ClaimsEscrow	returnEth	-	-	44163	2	-
ClaimsEscrow	safeTransfer	-	-	86071	2	-
ClaimsEscrow	setCooldownPeriod	-	-	29735	2	-
ClaimsEscrow	setPendingGovernance	-	-	47169	3	-
ClaimsEscrow	transfer	-	-	83251	2	-
ClaimsEscrow	withdrawClaimsPayout	73488	98996	75108	62	-
CpFarm	acceptGovernance	-	-	28217	3	-
CpFarm	compoundRewards	34569	337959	186264	4	-
CpFarm	depositCp	-	-	78870	4	-
CpFarm	depositCpSigned	-	-	113114	2	-
CpFarm	depositEth	96350	162010	138556	17	-
CpFarm	setEnd	-	-	37082	1	-
CpFarm	setPendingGovernance	-	-	47180	3	-
CpFarm	updateFarm	23417	30622	28215	3	-
CpFarm	withdrawCp	80947	155091	121136	7	-
CpFarm	withdrawRewards	44964	151471	94109	5	-
CpFarm	withdrawRewardsForUser	-	-	47689	1	-
ERC20	approve	29356	53935	49082	44	-
ERC20	transfer	-	-	51545	1	-
LpAppraisor	acceptGovernance	-	-	28172	3	-
LpAppraisor	setCurve	-	-	34284	1	-
LpAppraisor	setPendingGovernance	-	-	47080	3	-
Master	acceptGovernance	-	-	28217	3	-
Master	massUpdateFarms	77589	224807	125538	5	-
Master	registerFarm	147497	360059	237047	21	-
Master	setAllocPoints	49318	210930	156654	18	-
Master	setPendingGovernance	-	-	47131	3	-
Master	setSolacePerBlock	141949	208406	170677	5	-
Master	withdrawRewards	188388	314504	251446	6	-
MockERC20	approve	46274	46586	46430	4	-
MockERC20	transfer	51481	51541	51530	11	-
MockProduct	_buyPolicy	244834	290546	275213	12	-
MockProduct	acceptGovernance	-	-	28229	3	-
MockProduct	addSigner	-	-	47322	3	-
MockProduct	buyPolicy	321087	359833	334002	6	-
MockProduct	cancelPolicy	108451	139104	129481	4	-
MockProduct	extendPolicy	98897	105851	101679	5	-
MockProduct	removeSigner	-	-	25406	2	-
MockProduct	setCoveredPlatform	-	-	30148	3	-
MockProduct	setMaxPeriod	-	-	29985	2	-
MockProduct	setMinPeriod	-	-	29999	2	-
MockProduct	setPaused	29792	29804	29798	12	-
MockProduct	setPendingGovernance	47148	47160	47152	3	-
MockProduct	setPolicyExpiration	50837	50873	50855	6	-
MockProduct	setPolicyManager	-	-	30148	3	-
MockProduct	updateActiveCoverAmount	28753	45853	34604	3	-
MockProduct	updateCoverAmount	116007	149233	135006	16	-
MockProduct	updatePolicy	128631	152853	141432	14	-
PolicyManager	acceptGovernance	-	-	28208	3	-
PolicyManager	addProduct	27419	91892	89949	128	-
PolicyManager	approve	-	-	48778	1	-
PolicyManager	burn	70031	81626	75829	4	-
PolicyManager	createPolicy	227787	256735	237554	9	-
PolicyManager	removeProduct	-	-	41648	2	-
PolicyManager	safeTransfer	-	-	88843	2	-
PolicyManager	setPendingGovernance	-	-	47210	3	-

PolicyManager	setPolicyDescriptor	-	-	47221	3	-
PolicyManager	setPolicyInfo	44223	49895	47059	2	-
PolicyManager	transfer	-	-	93596	2	-
PolicyManager	transferFrom	82166	94008	86528	21	-
PolicyManager	updateActivePolicies	-	-	156184	1	-
Registry	acceptGovernance	-	-	28185	3	-
Registry	setClaimsEscrow	30076	47176	46740	118	-
Registry	setLocker	-	-	47177	2	-
Registry	setMaster	-	-	47176	4	-
Registry	setPendingGovernance	-	-	47109	3	-
Registry	setPolicyManager	27253	47153	46712	123	-
Registry	setRiskManager	47163	47175	47174	82	-
Registry	setSolace	47185	47197	47191	4	-
Registry	setTreasury	30052	47152	45015	8	-
Registry	setVault	47185	47197	47196	121	-
Registry	setWeth	47151	47175	47174	122	-
RiskManager	acceptGovernance	-	-	28185	3	-
RiskManager	addProduct	36985	121027	81214	10	-
RiskManager	removeProduct	26145	49846	39690	8	-
RiskManager	setPartialReservesFactor	-	-	29841	5	-
RiskManager	setPendingGovernance	-	-	47115	3	-
RiskManager	setProductParams	30146	195063	129673	11	-
SOLACE	acceptGovernance	-	-	28195	6	-
SOLACE	addMinter	27417	47317	29635	9	-
SOLACE	approve	46297	46609	46533	13	-
SOLACE	mint	38720	72932	55836	20	-
SOLACE	removeMinter	-	-	25355	2	-
SOLACE	setMaxSupply	31901	31913	31907	4	-
SOLACE	setPendingGovernance	-	-	47176	6	-
SOLACE	transfer	28770	51542	39144	34	-
SolaceEthLpFarm	acceptGovernance	-	-	28217	3	-
SolaceEthLpFarm	depositLp	379342	598984	515548	16	-
SolaceEthLpFarm	depositLpSigned	-	-	436350	2	-
SolaceEthLpFarm	mintAndDeposit	-	-	712118	3	-
SolaceEthLpFarm	setAppraisor	-	-	28993	1	-
SolaceEthLpFarm	setEnd	52490	57837	55164	2	-
SolaceEthLpFarm	setPendingGovernance	-	-	47171	3	-
SolaceEthLpFarm	updateFarm	23417	130141	73344	5	-
SolaceEthLpFarm	withdrawLp	185169	262941	215135	15	-
SolaceEthLpFarm	withdrawRewards	55622	145048	100967	5	-
SolaceEthLpFarm	withdrawRewardsForUser	-	-	58359	1	-
Treasury	acceptGovernance	-	-	28195	3	-
Treasury	refund	54039	96405	73800	3	-
Treasury	routePremiums	28928	159968	60036	11	-
Treasury	setPendingGovernance	-	-	47103	3	-
Treasury	setPremiumRecipients	58683	473931	146685	6	-
Treasury	spend	38055	62600	54418	6	-
Treasury	unwrap	-	-	49447	1	-
Treasury	withdraw	25928	40231	35463	3	-
Treasury	wrap	-	-	49122	1	-
Vault	acceptGovernance	-	-	28239	2	-
Vault	approve	-	-	46241	3	-
Vault	depositEth	50303	84495	78050	116	-
Vault	depositWeth	78203	120059	109063	19	-
Vault	increaseAllowance	-	-	46455	3	-
Vault	requestEth	27386	49048	35132	25	-
Vault	setCooldownWindow	30466	30502	30494	13	-
Vault	setEmergencyShutdown	25037	46949	44514	9	-
Vault	setPendingGovernance	-	-	47192	3	-
Vault	setRequestor	25878	47790	43747	65	-
Vault	startCooldown	-	-	44540	19	-
Vault	stopCooldown	-	-	22639	2	-
Vault	transfer	37902	55814	43076	13	-
Vault	transferFrom	51917	58790	56482	3	-
Vault	withdrawEth	43024	82446	71054	101	-
Vault	withdrawWeth	50905	106525	64628	16	-
WaaveProduct	addSigner	-	-	47322	1	-
WaaveProduct	buyPolicy	319439	370523	336512	12	-
WaaveProduct	setCoveredPlatform	-	-	35244	2	-
WaaveProduct	submitClaim	281624	291448	284080	16	-

```
·····················|·····························|·············|·············|·············|···············|·············
|  WETH9             ·  approve                    ·      46275  ·      46587  ·      46358  ·           37  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  WETH9             ·  deposit                    ·      34867  ·      69067  ·      59972  ·           41  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  WETH9             ·  transfer                   ·          -  ·          -  ·      51481  ·            1  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  WETH9             ·  withdraw                   ·          -  ·          -  ·      42032  ·            1  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  Deployments                                     ·                                        ·  % of limit   ·             |
·····················|·····························|·············|·············|·············|···············|·············
|  AaveV2Product                                   ·          -  ·          -  ·    3757723  ·       12.5 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  ClaimsEscrow                                    ·    2797211  ·    2797223  ·    2797223  ·        9.3 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  CpFarm                                          ·    1886666  ·    1926572  ·    1924432  ·        6.4 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  GasGriefer                                      ·          -  ·          -  ·     109549  ·        0.4 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  LpAppraisor                                     ·     608326  ·     608338  ·     608337  ·          2 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  Master                                          ·     756197  ·     756281  ·     756278  ·        2.5 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  MockAToken                                      ·          -  ·          -  ·      78931  ·        0.3 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  MockERC20                                       ·          -  ·          -  ·     671228  ·        2.2 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  MockProduct                                     ·    3805979  ·    3806015  ·    3806007  ·       12.7 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  PolicyDescriptor                                ·          -  ·          -  ·     272373  ·        0.9 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  PolicyManager                                   ·          -  ·          -  ·    3056070  ·       10.2 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  Registry                                        ·          -  ·          -  ·     794289  ·        2.6 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  RiskManager                                     ·    1551693  ·    1551705  ·    1551705  ·        5.2 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  SOLACE                                          ·          -  ·          -  ·    1391215  ·        4.6 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  SolaceEthLpFarm                                 ·    3711372  ·    3751290  ·    3748256  ·       12.5 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  Treasury                                        ·    1598857  ·    1712932  ·    1627894  ·        5.4 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  Vault                                           ·    2794821  ·    2794833  ·    2794833  ·        9.3 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  WaaveProduct                                    ·          -  ·          -  ·    3714975  ·       12.4 %  ·          -  |
·····················|·····························|·············|·············|·············|···············|·············
|  WETH9                                           ·          -  ·          -  ·     827236  ·        2.8 %  ·          -  |
·--------------------------------------------------|-------------|-------------|-------------|---------------|-------------·

  533 passing (15m)
  2 failing

  1) AaveV2ProductKovan
       submitClaim
         should support all atokens:

      AssertionError: expected '19/21 supported atokens' to equal '21/21 supported atokens'
      + expected - actual

      -19/21 supported atokens
      +21/21 supported atokens

      at Context.<anonymous> (test/AaveV2ProductKovan.test.ts:430:69)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:97:5)
      at runNextTicks (internal/process/task_queues.js:66:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)

  2) PolicyManager
       policies
         can get policy info:

      AssertionError: expected false to equal true
      + expected - actual

      -false
      +true

      at Context.<anonymous> (test/PolicyManager.test.ts:199:56)
      at runMicrotasks (<anonymous>)
      at processTicksAndRejections (internal/process/task_queues.js:97:5)
      at runNextTicks (internal/process/task_queues.js:66:3)
      at listOnTimeout (internal/timers.js:523:9)
      at processTimers (internal/timers.js:497:7)
```

Code Coverage

Quantstamp was unable to compute the test coverage for this project.

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

```
07df3112c98abcea9328f0f0526a855f58e8c3c64f8a0bfc48e68f5f947fafda ./contracts/ClaimsEscrow.sol
f050339746949810cd60fede43227ffd795988562212b51f008a0e846959dc0b ./contracts/Governable.sol
233b8ee27bc32efbd5c6b92a5521a9a1cabb89f666842218eac220dc79ee6153 ./contracts/PolicyManager.sol
e56c40067b73405db5e4c588c7befbbce872952dd59709b6a74807f0508d8cbf ./contracts/Registry.sol
ce9eca9555b7ac3af94a98e088c8b47b87b1c83fa0e2f0722e53bf8d520420b3 ./contracts/RiskManager.sol
99af2df732695a992f1846917a51a09b1924ae883b574cd3ef06cffb7e796802 ./contracts/SOLACE.sol
5203b52ec6036a1a3592297979324e9204155088394a4da8c8b59677f9932ff6 ./contracts/Treasury.sol
f2b37f9918f8928d634c6592e4a80f9f445f75c33b9a40435a56aa4fccf05388 ./contracts/Vault.sol
b24dbf5a00e1652e08121bc0dea595377ed43172d87bde5960759167470fa657 ./contracts/products/BaseProduct.sol
```

Tests

```
7af32d2d02084fa40469849e43949b77434099cd7588545df163aedf477790d3 ./test/AaveV2Product.test.ts
c6299c10e7469efc9beb71780fdd26e18205c4ceff5979ebc0531bc4f6f4df7e ./test/AaveV2ProductKovan.test.ts
c071dc2661a70f1b33f4783a2bd3ec7c4dc366f071ccc961301a0c98ec2ac6e2 ./test/BaseProduct.test.ts
52fc3a7ebc16b76d4978736e0c07aceaf8c5ae9f815325eb96f447e6cbe04b12 ./test/ClaimsEscrow.test.ts
ddbea6c28a1647eec79f32fbac10d0a0f0d7558933f4bf32cce475ed850a376c ./test/CompoundProduct.test.ts
d9d67eb0a6f59ab23bdf283e4128eb0fff988c38ae9529933bcd2a3a4ccdd156 ./test/CompoundProductRinkeby.test.ts
305ed5267eaef4661faead296cf2da113bd3c548f8eedfde93c739ebfb0df7d1 ./test/CpFarm.test.ts
b09534df1788d8e0880d028210ee0c825b6e10a585c0f23f7f9422f90370ab91 ./test/CurveProduct.test.ts
24affdf15dbabc74d85a313ab7622ba75a454f5aba298f290c92c82ea2e737c3 ./test/Governable.test.ts
35eca33b32b0ebabfab556dd1b76a9710826b20398fc00089e1c92d90fa3679b ./test/LpAppraisor.test.ts
97c6724ddfe2b6065a5aec7db5c8a0ebe912fb2b34c23b7444a1b76e31cabcb2 ./test/Master.test.ts
08f7d96c66a04ae7646e58c93941eb4db3f046eb37d24e906d55693235b7f03f ./test/PolicyManager.test.ts
9fb27c1f3684ce6aee587cdbe8cdc5078158c7d532d0b254250c203925692445 ./test/Registry.test.ts
55b6f1976824095e308e8e0eb88dbf36e7f522ec6f6aa9bc9768d6db4e27c538 ./test/RiskManager.test.ts
9207d9035d39d09a49ef318b7be201069909d958d77c7019dab4f2d4b875b146 ./test/SOLACE.test.ts
568b06bbdb82ca1de5d2cee02fce94a33b6b57445d6faff40b988bd8472cb975 ./test/SolaceEthLpFarm.test.ts
f4e5ad03db1ceb6c49d094f637102930debe35bc13881f67b6f29dc9340fe577 ./test/Treasury.test.ts
447791fc699781c375c63ad91382c989a75465c72fdbf4acf0055da482202fcb ./test/Vault.test.ts
8fbf1eb6ebac99b50e1bf078ca5b00f684be0094c8580b1b114a8efbb91e9169 ./test/WaaveProductKovan.test.ts
250311e1ff6cb7569ec213409ac35ba7693e355a33f8eb5b78ff29ae0d0f5fac ./test/WaaveProductRinkeby.test.ts
83cbe9552b63efe8eba7dfdb57e650d551498104ef5db316c4d73cc6016605ef ./test/YearnV2Product.test.ts
08cdd06149a848c48930fc4330d31a325115d7f0beaaddf8f12c6da7ad32a040 ./test/utilities/artifact_importer.ts
545b5997743b3eff94360e77b841bfce06554a12ee159a53f0dd60f239151d62 ./test/utilities/getPermitNFTSignature.ts
59c42b677ed94f1ac4b7051010f4ac51feb089a0f2ed116498a61ec3cebb27ce ./test/utilities/math.ts
66163d4b17a8bfbc387f9cd14845b4601548b288bde702a39fcbc0f40bb891c2 ./test/utilities/path.ts
435af79a85d6448ec42245e150b790a32d51ffeacd0c70fd22022953650aff5a ./test/utilities/positionDescription.ts
6683a0b8d176c206b9312a75b9975c6a9d8a1a13eb50a9d0f7add2247bd0d3de ./test/utilities/setStorage.ts
0575da87030a1f097f6814146c13f4e34710f19a2030524448abe49c0e4d9e47 ./test/utilities/signature.ts
0ad0d77d7a6d5e5b9a75f743bddc554b70ddb69c10ca66ef82537f7cae1448ca ./test/utilities/time.ts
b3ee22ad9331f4b96631dda84b635243af27e0d04fee0301116b9973246196d6 ./test/utilities/uniswap.ts
```

Changelog

- 2021-10-07 - Initial report [89f82de]
- 2021-10-18 - Revised report [928d61d]

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.