



November 29th 2022 — Quantstamp Verified

# Metronome Synth

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

## Executive Summary

Type	Defi Synthetic Assets Lending Platform						
Auditors	Poming Lee, Senior Research Engineer Roman Rohleder, Research Engineer Mostafa Yassin, Security Engineer Setareh Ghorshi, Auditing Engineer I						
Timeline	2022-10-25 through 2022-11-29						
Languages	Solidity						
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review						
Specification	<a href="#">Vesper Synth-High level doc</a>						
Documentation Quality	<div><div></div><div></div></div> Medium						
Test Quality	<div><div></div><div></div></div> Medium						
Source Code	<table><tr><th>Repository</th><th>Commit</th></tr><tr><td><a href="#">blogpriv/metronome-synth</a></td><td>5108b86 initial audit</td></tr><tr><td><a href="#">blogpriv/metronome-synth/d74e64ccd7915ec5a85f9eecd80a5bcc04fc66c6</a></td><td>d74e64c fixes</td></tr></table>	Repository	Commit	<a href="#">blogpriv/metronome-synth</a>	5108b86 initial audit	<a href="#">blogpriv/metronome-synth/d74e64ccd7915ec5a85f9eecd80a5bcc04fc66c6</a>	d74e64c fixes
Repository	Commit						
<a href="#">blogpriv/metronome-synth</a>	5108b86 initial audit						
<a href="#">blogpriv/metronome-synth/d74e64ccd7915ec5a85f9eecd80a5bcc04fc66c6</a>	d74e64c fixes						

Total Issues	16 (9 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	2 (1 Resolved)
Low Risk Issues	3 (2 Resolved)
Informational Risk Issues	11 (6 Resolved)
Undetermined Risk Issues	0 (0 Resolved)



⚠ High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
⚠ Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
✓ Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
ℳ Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
❓ Undetermined	The impact of the issue is uncertain.
⬤ Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
⬢ Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
⬢ Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
⬢ Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

## Summary of Findings

The current project is a DeFi lending protocol for synthetic assets. Quantstamp has, on a best-effort basis, with 4 auditors working independently (and later on syncing on their findings), identified 16 issues of various levels of severity. This report also lists several best practice recommendations, in addition to concerns regarding documentation. Furthermore, Quantstamp recommends adding additional tests to increase the branch coverage to at least 90%, in order to avoid functional bugs that are not necessarily security issues. We highly recommend addressing the findings before going live.

**2022-12-15 Update:** During this fix-check, the dev team has brought some of the statuses of findings either into fixed whereas the status of the others remains unchanged. New code is introduced in the given final commit hash, mostly due to refactoring. It is worth mentioning that all these changes unrelated to the findings in this report are not part of the audit. In addition, an issue that `rewardsDistributors` cannot be removed has been reported and the fix (in commit hash `2b70235`) to this issue has been checked by Quantstamp.

ID	Description	Severity	Status
QSP-1	There Is No Backup Oracle nor Protection From Erroneous Price Data	^ Medium	Unresolved
QSP-2	Non-Restricted Input Validation for Protocol Parameters	^ Medium	Mitigated
QSP-3	<code>maxTotalSupply()</code> Could Be Smaller than <code>totalSupply_</code>	^ Low	Acknowledged
QSP-4	Missing Input Checks	^ Low	Mitigated
QSP-5	<code>DebtToken.SECONDS_PER_YEAR</code> Does Not Take Into Account Leap Years	^ Low	Fixed
QSP-6	Missing Initializer Calls	o Informational	Fixed
QSP-7	Transaction Ordering Dependence for <code>initialize()</code> Functions	o Informational	Acknowledged
QSP-8	Upgradable Proxy Contracts	o Informational	Unresolved
QSP-9	Privileged Roles and Ownership	o Informational	Unresolved
QSP-10	Sandwich Attack in <code>swap()</code>	o Informational	Fixed
QSP-11	Block Timestamp Manipulation	o Informational	Unresolved
QSP-12	Allowance Double-Spend Exploit	o Informational	Mitigated
QSP-13	Application Monitoring Can Be Improved by Emitting More Events	o Informational	Fixed
QSP-14	Clone-and-Own	o Informational	Acknowledged
QSP-15	Unlocked Pragma ( <a href="#">SWC-103</a> )	o Informational	Fixed
QSP-16	Unlocked Versions in <code>package.json</code>	o Informational	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:**

1. If the final commit hash provided by the client contains features that are not within the scope of the audit or an associated fix review, those features are excluded from consideration in this report.
2. The following files have been requested to exclude from the scope of the audit:

- contracts/dependencies/\*.sol
- contracts/access/\*.sol
- contracts/mock/\*.sol
- contracts/interfaces/\*.sol

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](#) v0.9.1

Steps taken to run the tools:

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

## Findings

### QSP-1 There Is No Backup Oracle nor Protection From Erroneous Price Data

**Severity:** *Medium Risk*

**Status:** Unresolved



**File(s) affected:** `contracts/Pool.sol` ,`contracts/DepositToken.sol` ,`contracts/DebtToken.sol`

**Description:** Price oracle is one of the core components of a DeFi platform. If a malicious hacker manipulates the price data, they could drain the fund from a pool in a very short period of time. So a mechanism to protect the platform from using manipulated or erroneous price data is often considered necessary for a DeFi platform. In the current project, `masterOracle` is used in all the critical valuation logic in the listed contracts for obtaining price data. The oracle that the current system uses does not have any backup currently. The system only collects price data of each token from one oracle so there is no backup oracle. The system could fail to work correctly when any of the oracles is operating abnormally or being manipulated.

**Recommendation:** Please add more than one oracle for each token in order to increase the security level. Also, consider adding some sanity checks to the collected price data.

**Update:** The dev team stated that "We will soon add another fallback oracle to resolve this issue."

## QSP-2 Non-Restricted Input Validation for Protocol Parameters

**Severity:** *Medium Risk*

**Status:** Mitigated

**File(s) affected:** `contracts/Pool.sol`

**Description:** Some of the functions are performing input validation, but the range of the values can be problematic for the functions of the protocol. In addition, multiple fee-related update functions in contract `Pool` are having no further constraints on the new fee value other than 100%. Consequently, users can be subject to very high fees. This can subject users to high fees with no prior announcements. The following instances have been noted:

- `updateDebtFloor()`: the `debtFloor` should have some form of boundaries as a high enough debt floor can cause a denial of service when people attempt to take/repay/liquidate a loan.
- `updateDepositFee()`: a `depositFee` equal to `1e18` means that all the deposits added by the user will be consumed as a fee.
- `updateLiquidatorLiquidationFee()`: a `liquidationFee` equal to `1e18` means that the entire underlying asset that can be liquidated will be sent to the liquidator. This can also cause DoS if the `liquidationFee` plus the `protocolFee` exceeds the account's underlying asset.
- `updateProtocolLiquidationFee()`: same as `liquidationFee`.
- `updateMaxLiquidable()`: `newMaxLiquidable_` has no cap and may be any value 0%-100%.
- `updateIssueFee()`: `_newIssueFee` has no cap and may be any value 0%-100%.
- `updateRepayFee()`: a `repayFee` of `1e18` means that the entire repayment amount is sent to the protocol.
- `updateSwapFee()`: a `SwapFee` of `1e18` means that the entire repayment amount is sent to the protocol.
- `updateWithdrawFee()`: a `withdrawFee` of `1e18` means that the entire repayment amount is sent to the protocol.

**Recommendation:** Consider adding hardcoded "sane" upper/lower bounds for these values, enforce this bound in said functions, and communicate this bound to users in public-facing documentation.

**Update:** `updateDebtFloor()` and `updateMaxLiquidable()` remain unchanged, while a `MAX_FEE_VALUE = 0.25e18; // 25%` limit has been applied to all other functions.

## QSP-3 `maxTotalSupply()` Could Be Smaller than `totalSupply_`

**Severity:** *Low Risk*

**Status:** Acknowledged

**File(s) affected:** `contracts/DebtToken.sol` ,`contracts/DepositToken.sol` ,`contracts/SyntheticToken.sol`

**Description:** The token contracts listed contain `updateMaxTotalSupply()` which set a `maxTotalSupply` smaller than `totalSupply_`. This should not happen, and when happened, would lead to edge cases that result in system inconsistencies.

**Recommendation:** Make sure that `newMaxTotalSupply_` in `updateMaxTotalSupply()` is always smaller than or equal to `totalSupply_`.

**Update:** The dev team stated that "Parameter gives us some flexibility regarding risk management, for instance msUSD supply is 100M and we want to force reduction to 90M without having to pause or shut down the Pool (impacting other features). The only inconsistency that we see is that `msUSD.totalSupply()` would be greater than `msUSD.maxTotalSupply()` but we don't think it's a problem." which means this is intended by design.

## QSP-4 Missing Input Checks

**Severity:** *Low Risk*

**Status:** Mitigated

**File(s) affected:** `contracts/DepositToken.sol` ,`contracts/SyntheticToken.sol` ,`contracts/DebtToken.sol` ,`contracts/Treasury.sol`

**Description:** It is important to validate inputs, even if they only come from trusted addresses, to avoid human error. The following functions do not have a proper validation of input parameters:

- `DepositToken.initialize()` does not check that parameter `_symbol` has a non-zero length, `decimals` and `_maxTotalSupplyInUsd` are non-zero.
- `DepositToken.withdraw()` should check that `to != 0x0`.
- `Treasury.pull()` should check that `to != 0x0`.
- `Treasury.migrateTo()` does not check that parameter `_newTreasury` is different from `0x0`.
- `DebtToken.initialize()` does not check that parameters `_interestRate` and `maxTotalSupply_` are non-zero or otherwise bound.
- `DebtToken._mint()` does not check that parameter `amount` is non-zero.
- `DebtToken._burn()` does not check that parameter `amount` is non-zero.
- `SyntheticToken.initialize()` does not check that parameter `_decimals` is non-zero or otherwise bound.
- `SyntheticToken._transfer()` does not check that parameter `amount` is non-zero.
- `SyntheticToken._mint()` does not check that parameter `amount` is non-zero.
- `SyntheticToken._burn()` does not check that parameter `amount` is non-zero.

12. `SyntheticToken._approve()` does not check that parameter `amount` is non-zero.

**Recommendation:** Perform input validations on these parameters.

**Update:** Some of the checks are added while there are still issues remaining, as listed below:

1. `DepositToken.initialize()` does not check that parameter `_maxTotalSupplyInUsd` is non-zero.
2. `DebtToken.initialize()` does not check that parameters `_interestRate` and `maxTotalSupply_` are non-zero or otherwise bound.
3. `DebtToken._mint()` does not check that parameter `amount` is non-zero.
4. `DebtToken._burn()` does not check that parameter `amount` is non-zero.
5. `SyntheticToken._transfer()` does not check that parameter `amount` is non-zero.
6. `SyntheticToken._mint()` does not check that parameter `amount` is non-zero.
7. `SyntheticToken._burn()` does not check that parameter `amount` is non-zero.
8. `SyntheticToken._approve()` does not check that parameter `amount` is non-zero.

## QSP-5 `DebtToken.SECONDS_PER_YEAR` Does Not Take Into Account Leap Years

**Severity:** *Low Risk*

**Status:** Fixed

**Description:** `DebtToken.SECONDS_PER_YEAR` assumes a constant 365 days per year. This does not hold for years with a leap day (Feb 29th). The calculated interest is therefore slightly off every four years.

**Recommendation:** Consider using a normalized value for the year days (i.e., 365.25) for precise computation of fees or clarify this limitation in user-facing documentation.

**Update:** Fixed based on the recommendation.

## QSP-6 Missing Initializer Calls

**Severity:** *Informational*

**Status:** Fixed

**File(s) affected:** `contracts/NativeTokenGateway.sol`

**Description:** Contracts following the [upgradeable pattern](#) implement initializer functions instead of constructors and need to call corresponding initializer functions of inherited contracts, if any are provided. Failing to do so may result in unexpected/inconsistent states after deployment. The following instances have been noted, where initializer calls are missing:

1. `NativeTokenGateway.constructor(): __ReentrancyGuard_init()`.

Note: In all above instances only calls to `__ReentrancyGuard_init()` are missing. While in the current implementation the functions and logic in `ReentrancyGuard.sol` still work without necessarily calling `__ReentrancyGuard_init()`, it is still advised to call the initializer function in case of future breaking changes.

**Recommendation:** Consider adding calls to the aforementioned initializer function.

**Update:** Fixed based on the recommendation.

## QSP-7 Transaction Ordering Dependence for `initialize()` Functions

**Severity:** *Informational*

**Status:** Acknowledged

**File(s) affected:** `contracts/Treasury.sol`, `contracts/DepositToken.sol`, `contracts/DebtToken.sol`, `contracts/PoolRegistry.sol`, `contracts/SyntheticToken.sol`, `contracts/RewardsDistribution.sol`

**Description:** The various `initialize()` functions of upgradeable contracts are not constructors. There’s a low-but-not-zero chance that someone can call these after the contracts have been deployed but before the development team calls them. Consequently, contracts may get initialized with values that are not desirable by the development team.

**Recommendation:** Be aware of this issue and be prepared to redeploy your contracts if these calls are front-run. Do not use your project until you have checked that your calls to these functions went through.

**Update:** The dev team stated that "Our deployment script deploys and initializes contracts atomically."

## QSP-8 Upgradeable Proxy Contracts

**Severity:** *Informational*

**Status:** Unresolved

**File(s) affected:** `contracts/*.sol`, `contracts/upgraders/*`

**Description:** All contracts in `contracts/*.sol`, except for `contracts/NativeTokenGateway.sol`, are designed to be upgradeable by using `contracts/upgraders/*`. This means that privileged users can change the logic of these contracts at any time at will. In another word, the code would change after the audit process. While upgradability is not a vulnerability in itself, token holders should be aware that the token contract can be upgraded at any given time. This audit does not guarantee the behavior of future contracts that the token may be upgraded to.

**Recommendation:** The fact that the contract can be upgraded and reasons for future upgrades should be communicated to users beforehand. Quantstamp recommends explicitly mentioning this information in a public document. Also, it is highly recommended to have all changes go through a full audit periodically.

## QSP-9 Privileged Roles and Ownership



Severity: *Informational*

Status: Unresolved

File(s) affected:

contracts/PoolRegistry.sol , contracts/Pausable.sol , contracts/Pool.sol , contracts/DepositToken.sol , contracts/RewardsDistributor.sol , contracts/Treasury.sol , contracts/DebtToken.sol , contracts/SyntheticToken.sol

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

The `PoolRegistry.sol` contract contains the following privileged roles:

- `governor`, as initialized during the `initialize()` execution to `msg.sender`:
  - . Assign a new `governor` address by calling `transferGovernorship()` (who in turn has to accept the role by calling `acceptGovernorship()`).
  - . Adding arbitrary non-zero addresses as pools by calling `registerPool()`.
  - . Remove any existing pool from the registry by calling `registerPool()`.

The `Pool` contract contains the following privileged roles:

- `governor`, as initialized during the `initialize()` execution to `msg.sender`:
  - . Assign a new `governor` address by calling `transferGovernorship()` (who in turn has to accept the role by calling `acceptGovernorship()`).
  - . Pause/Unpause the contract at any point (However this functionality does not impact any function calls in the given version) by calling `pause()/unpause()`.
  - . Shutdown/Open the contract at any point (thereby prohibiting/re-enabling calls to `liquidate()` and `swap()`) by calling `shutdown()/open()`.
- 

The `DepositToken.sol` contract contains the following privileged roles:

- `governor`, as initialized during the `initialize()` execution to `msg.sender`:
  - . Assign a new `governor` address by calling `transferGovernorship()` (who in turn has to accept the role by calling `acceptGovernorship()`).
  - . Pause/Unpause the contract at any point (thereby prohibiting/re-enabling calls to `deposit()`) by calling `pause()/unpause()`.
  - . Shutdown/Open the contract at any point (thereby prohibiting/re-enabling calls to `withdraw()`) by calling `shutdown()/open()`.
  - . Change the current collateralization ratio (**From 0% to up to 100%**) and thereby impact the amount of underlying one gets when withdrawing or amount of `msdTOKEN` when one deposits by calling `updateCollateralizationRatio()`.
  - . Change the current minimum deposit time (**from zero to practically unlimited deposit time**) and thereby impact how long one needs to wait until withdrawal is possible by calling `updateMinDepositTime()`.
  - . Change the current maximum total supply of `msdTOKEN` (**to any arbitrary value including a smaller maximum than before**) and thereby impact the ability to mint or depositing by calling `updateMaxTotalSupplyInUsd()`.
  - . Enable/Disable the deposit token and thereby impacting the ability to mint or depositing by calling `toggleIsActive()`.
- `pool`, as enforced through modifier `onlyPool` and initialized during `initialize()` to `_pool` or updated through `setPool()` by the current `governor`:
  - . Mint an arbitrary amount of `msdTOKEN` tokens (up to `maxTotalSupplyInUsd`) to an arbitrary address by calling `mint()`.
  - . Burn an arbitrary amount of `msdTOKEN` tokens from any account (unrestricted from any `pause` states) by calling `burn()`.
  - . Transfer an arbitrary amount of `msdTOKEN` of any account to any other account by calling `seize()`.

The `RewardsDistributor.sol` contract contains the following privileged roles:

- `governor`, as initialized during the `initialize()` execution to `msg.sender`:
  - . Assign a new `governor` address by calling `transferGovernorship()` (who in turn has to accept the role by calling `acceptGovernorship()`).
  - . Change the reward token speens by calling `updateTokenSpeed()/updateTokenSpeeds()`.

The `Treasury.sol` contract contains the following privileged roles:

- `pool`, as enforced through modifier `onlyPool` and initialized during `initialize()` to `_pool` or updated through `setPool()` by the current `governor`:
  - . **Move all deposit, synthetic and underlying tokens from the current contract to an arbitrary other address** by calling `migrateTo()`.

The `DebtToken.sol` contract contains the following privileged roles:

- `governor`, as initialized during the `initialize()` execution to `msg.sender`:
  - . Assign a new `governor` address by calling `transferGovernorship()` (who in turn has to accept the role by calling `acceptGovernorship()`).
  - . Change the maximum total supply and thereby impacting the amount issuable debt tokens by calling `updateMaxTotalSupplyInUsd()`.
  - . **Change the interest rate to an arbitrary value** by calling `updateInterestRate()`.
  - . Deactivate/Activate the debt token (and thereby disable/enable the ability to perform swaps via `Pool.swap()`) by calling `toggleIsActive()`.
- `syntheticToken`, as enforced through modifier `onlyIfSyntheticToken` and initialized during `initialize()` to `_syntheticToken`:
  - . Mint new debt tokens up to `maxTotalSupplyInUsd` to an arbitrary address by calling `mint()`.

The `SyntheticToken.sol` contract contains the following privileged roles:

- `governor`, as initialized during the `initialize()` execution to `msg.sender`:
  - . Assign a new `governor` address by calling `transferGovernorship()` (who in turn has to accept the role by calling `acceptGovernorship()`).
  - . Deactivate/Activate the synthetic token (and thereby disable/enable the ability of issuing debt/synthetic tokens via `DebtToken.issue()`) by calling `toggleIsActive()`.
- Any registered pool as per `poolRegistry.poolExists()` or any registered debt token of any pool:
  - . **Mint an arbitrary amount of synthetic tokens to an arbitrary address** by calling `mint()`.
  - . Burn an arbitrary amount of synthetic tokens of an arbitrary address by calling `burn()`.
  - . **Arbitrarily move synthetic tokens from one account to another** by calling `seize()`.

The `UpgraderBase.sol` contract contains the following privileged roles:

- `owner`, as initialized during the `constructor()` execution to `msg.sender`:
  - . Assign a new `owner` address by calling `transferOwnership()`.
  - . Renounce the role (**and thereby preventing any future calls to the followingly listed funcitons!**) by calling `renounceOwnership()`.
  - . Upgrade a given proxy implementation by calling `upgrade()`.
  - . Upgrade a given proxy implementation and execute a function by calling `upgradeAndCall()`.

**Recommendation:** Clarify the impact of these privileged actions to the end-users via publicly facing documentation.

## QSP-10 Sandwich Attack in `swap()`

**Severity:** *Informational*

**Status:** Fixed

**Description:** The `swap` feature is susceptible to sandwich attacks. An adversary can take advantage of the situation by front-running a large swap transaction in order to take advantage of the tokens' price change after the swap.

**Exploit Scenario:**

1. Alice tries to swap a large amount of token `a` with token `b`. This will result in an increase in the price of `a` and a decrease in the price of `b`.
2. The adversary who holds token `b` can front run this transaction by swapping their `b` tokens with `a`. This will result in an increase in the price of `b` and a decrease in the price of `a`. Therefore, Alice will receive less amount of token `b` compared to before.
3. After Alice's transaction, the adversary can swap their `a` tokens back to `b`, making a profit since the price of `a` is higher now.

**Recommendation:** This issue is the result of how the network works so there is no clear solution for it. Nevertheless, it is good to be aware of such scenarios.

**Update:** It is confirmed that the swap feature in the platform is slippage free.

## QSP-11 Block Timestamp Manipulation

**Severity:** *Informational*

**Status:** Unresolved

**File(s) affected:** `contracts/DebtToken.sol`, `contracts/RewardsDistributor.sol`

**Description:** Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes. If a smart contract relies on a timestamp, it must take this into account.

The following instances making use of `block.timestamp` have been noted:

1. `DebtToken.sol` #L91.
2. `DebtToken.sol` #L108.
3. `DebtToken.sol` #L112.
4. `DebtToken.sol` #L329.
5. `DebtToken.sol` #L330.
6. `RewardsDistributor.sol` #L179.
7. `RewardsDistributor.sol` #L185.
8. `RewardsDistributor.sol` #L187.
9. `RewardsDistributor.sol` #L206.
10. `RewardsDistributor.sol` #L210.

These rely on `block.timestamp` to be correct, but that aspect can be manipulated by validators/attackers for their purposes by up to 900 seconds.

**Recommendation:** Clarify the impact in the given protocol design in user-facing documentation and, if necessary, use an oracle for time inquiries.

## QSP-12 Allowance Double-Spend Exploit

**Severity:** *Informational*

**Status:** Mitigated

**File(s) affected:** `contracts/DepositToken.sol`, `contracts/SyntheticToken.sol`

**Description:** As it presently is constructed, the contract is vulnerable to the [allowance double-spend exploit](#), as with other ERC20 tokens. A similar problem occurs in `ERC20/ERC20Permit.sol` in functions `permit()` and `transferFrom()`

**Exploit Scenario:**

1. Alice allows Bob to transfer `N` amount of Alice's tokens ( $N > 0$ ) by calling the `authorizeOperator()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` ( $M > 0$ ) the number of Alice's tokens Bob is allowed to transfer, so she calls the `authorizeOperator()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transfer()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain the ability to transfer another `M` tokens



5. Before Alice notices any irregularities, Bob calls the `transfer()` method again, this time to transfer `M` Alice's tokens.

**Recommendation:** The exploit (as described above) can be mitigated through the use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`. Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on a traditional `approve()` / `transferFrom()` should keep in mind that they have to set allowance to `0` first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contracts.

## QSP-13 Application Monitoring Can Be Improved by Emitting More Events

**Severity:** *Informational*

**Status:** Fixed

**File(s) affected:** `contracts/RewardsDistributor.sol`

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

1. `RewardsDistributor._updateTokenIndex(): tokenStates[]`.

**Recommendation:** Consider emitting the events.

**Update:** Fixed based on the recommendation.

## QSP-14 Clone-and-Own

**Severity:** *Informational*

**Status:** Acknowledged

**File(s) affected:** `contracts/dependencies/*`

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

**Recommendation:** Rather than the clone-and-own approach, a good industry practice is to use the Hardhat framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries. Specifically, it is recommended to use OpenZeppelin's [npm](#) package to import the smart contracts that were cloned and not modified.

For the cloned and modified smart contracts, consider if the modifications were necessary. If they were, break out the cloned contracts into separate files and add comments at the top of the files linking to the tagged OpenZeppelin repository file from which the local file was cloned. Add comments specifying the changes made locally to the cloned OpenZeppelin files.

**Update:** The dev team stated that "The contracts under `/dependencies/` folder didn't change. We have it here to ensure that we won't break upgradable contracts by accidentally using new versions."

## QSP-15 Unlocked Pragma ([SWC-103](#))

**Severity:** *Informational*

**Status:** Fixed

**File(s) affected:** `contracts/lib/MappedEnumerableSet.sol`

**Description:** Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version and above, hence the term "unlocked".

**Recommendation:** For consistency and to prevent unexpected behavior in the future, it is recommended to remove the caret to lock the file onto a specific Solidity version.

**Update:** Fixed based on the recommendation.

## QSP-16 Unlocked Versions in `package.json`

**Severity:** *Informational*

**Status:** Fixed

**File(s) affected:** `./package.json`

**Description:** "Unlocked" versions may make build environments not fully reproducible, and also carry the risks of unintentionally breaking or introducing security issues.

**Recommendation:** We recommend locking the versions of dependencies.

**Update:** Fixed based on the recommendation.

## Automated Analyses

Slither

Slither (v0.9.1) reported [329](#) results, all of which were either identified as false positives or included in the findings of this report.

## Adherence to Specification



1. The difference between the system being paused and shut down is unclear. It can be further explained in the documentation.

2. The following typographical errors have been noted:

1. `DepositTokenStorage.sol#L20`: `de` -> `the`.

2. `Pool.sol#L169`: `Get if` -> `Returns whether`.

3. `RewardsDistributor.sol#L203`: `token token` -> `token`.

4. `DebtToken.sol#L234`: `_feeAmount` -> `_repayFee`.

3. Several external/public functions are missing proper NatSpec comments. We recommend adding these to improve code documentation.
- ## Adherence to Best Practices
1. Custom errors in Solidity are more gas-efficient than using `require` statements with an error string, see: [custom-errors](#) for more details.

2. Function state mutability can be restricted to pure: `DebtToken.approve()`

3. Function state mutability can be restricted to pure: `DebtToken.transfer()`

4. Function state mutability can be restricted to pure: `DebtToken.transferFrom()`

5. Consider adding a check that can detect the change of `governor` during a proxy upgrade for `contracts/upgraders/*` since this variable is critical.

6. For `contracts/Pool.sol`: naming is grammatically incorrect in cases such as `isSyntheticTokenExists`, `isDepositTokenExists`, and `isDebtTokenExists`. The names can be changed into `doesSyntheticTokenExist`, `doesDepositTokenExist`, and `doesDebtTokenExist`.

7. For `contracts/SyntheticToken.sol`, `contracts/DepositToken.sol`, and `contracts/DebtToken.sol`: `onlyIfCanSeize` and `onlyIfCanBurn` can be combined into one modifier, `onlyIfIsPool`, since they both check if `msg.sender` is the pool contract.

8. To facilitate logging it is recommended to index address parameters within events. Therefore the `indexed` keyword should be added to the (other) address parameters in

1. `PoolRegistry.PoolRegistered()`,

2. `PoolRegistry.PoolUnregistered()`.

3. `Pausable.Paused()`.

4. `Pausable.Shutdown()`.

5. `Pausable.Unpaused()`.

6. `Pausable.Open()`.

7. `Pool.RewardsDistributorAdded()`.

8. `RewardsDistributor.RewardClaimed()`.

9. For improved readability and code quality it is advised to remove duplicate or unused code. In this regard consider the following cases:

1. Modifier `onlyIfAuthorized` is never used in contract `DebtToken.sol` and may therefore be removed.

2. State variable `lastDepositOf` in `DepositTokenStorage.sol` remains unused (the surrounding code comment also refers to a non-existent variable `minDepositTime`).

10. To improve readability and lower the risk of introducing errors when making code changes, it is advised to not use magic constants throughout code, but instead declare them once (as constant and commented) and use these constant variables instead. Following instances should therefore be changed accordingly:

1. `DebtToken.sol#L235`: `1e18`.
- ## Test Results
- ### Test Suite Results
- All tests have passed.
- ```
DebtToken
  default values
  issue
    should not revert if paused (678ms)
    should revert if shutdown
    should revert if surpass max supply in usd (75ms)
    should revert if synthetic does not exist (250ms)
    should revert if synthetic is not active
    should revert if debt token is not active
    should revert if user1 has not enough collateral deposited
    should revert if amount to issue is 0
    should revert if new debt < debt floor (58ms)
    should issue msAsset (issueFee == 0) (3701ms)
    should issue msAsset (issueFee > 0) (3933ms)
    should issue max issuable amount (issueFee == 0) (1036ms)
    should issue max issuable amount (issueFee > 0) (1756ms)
  when user1 issue some msETH
    repay
      should not revert if paused (606ms)
      should revert if shutdown
      should revert if amount is 0
      should revert if amount > unlocked collateral amount (44ms)
      should revert if new debt < debt floor (59ms)
      should allow repay if new debt == 0 (824ms)
      should allow repay if new debt > debt floor (825ms)
      should repay all debt (repayFee == 0) (798ms)
      should repay if amount < debt (repayFee == 0) (793ms)
      should repay if amount == debt (repayFee > 0) (1280ms)
      should repay if amount < debt (repayFee > 0) (1373ms)
      should repay all debt (repayFee > 0) (3832ms)
    repayAll
      should not revert if paused (767ms)
      should revert if shutdown
      should repay all debt (repayFee == 0) (1055ms)
      should repay all debt (repayFee > 0) (4351ms)
  when some synth was issued
    burn
      should burn (473ms)
      should revert if not authorized
      should not add address(0) to the users array (51ms)
      should remove debt token from user1 array only if burning all (126ms)
      should trigger rewards update (344ms)
```

```
transfer
  should revert when transferring
transferFrom
  should revert when transferring
allowance
  should revert when calling allowance
approve
  should revert when approving
balanceOf & totalSupply - get updated values without calling accrueInterest()
  should get updated balance (82ms)
  should not accrue interest if rate is 0
  should accrue interest after changing interest rate (344ms)
  should stop accruing interest after changing interest rate to 0 (1090ms)
accrueInterest
  should accrue interest (1122ms)
  should not accrue interest if rate is 0 (58ms)
  should accrue interest after changing interest rate (2347ms)
  should stop accruing interest after changing interest rate to 0 (2393ms)
  should not accrue interest backwards after changing interest rate from 0 (1160ms)
  should mint accrued fee to feeCollector (1232ms)
updateMaxTotalSupply
  should update collateral factor (89ms)
  should revert if using the current value (38ms)
  should revert if not governor
updateInterestRate
  should update interest rate (99ms)
  should revert if using the current value (43ms)
  should revert if not governor
toggleIsActive
  should toggle isActive flag (93ms)
  should revert if not governor

Deployments
(node:1613) ExperimentalWarning: stream/web is an experimental feature. This feature could change at any time
(Use `node --trace-warnings ...` to show where the warning was created)
Pool
  should have correct params
  should upgrade implementation (937ms)
  should fail if implementation breaks storage (87ms)
Treasury
  should have correct params
  should upgrade implementation (83ms)
NativeTokenGateway
  should have correct params
DepositToken
  should have the same proxy admin
  should have the same implementation
USDC DepositToken
  token should have correct params
  should upgrade implementation (303ms)
  should fail if implementation breaks storage (72ms)
WAVAX DepositToken
  token should have correct params
  should upgrade implementation (269ms)
  should fail if implementation breaks storage (72ms)
SyntheticToken
  should have the same proxy admin
  should have the same implementation
msBTC SyntheticToken
  token should have correct params
  should upgrade implementation (209ms)
  should fail if implementation breaks storage (69ms)
msUSD SyntheticToken
  msUSD token should have correct params
  should upgrade implementation (217ms)
  should fail if implementation breaks storage (69ms)
DebtToken
msBTC DebtToken
  token should have correct params
  should upgrade implementation (322ms)
  should fail if implementation breaks storage (86ms)
msUSD DebtToken
  token should have correct params
  should upgrade implementation (323ms)
  should fail if implementation breaks storage (77ms)
PoolRegistry
  should have correct params
  should upgrade implementation (114ms)

DepositToken
when user has some balance
  withdraw
    should revert not if paused (376ms)
    should revert if shutdown
    should revert if amount is 0
    should revert if amount > unlocked collateral amount
    should withdraw if amount <= unlocked collateral amount (withdrawFee == 0) (450ms)
    should withdraw if amount <= unlocked collateral amount (withdrawFee > 0) (602ms)
    should withdraw collateral to another user (512ms)
    should trigger rewards update (456ms)
  deposit
    should revert if paused
    should revert if shutdown
    should revert if surpass max supply in usd (54ms)
    should revert if collateral amount is 0
    should revert if MET balance is not enough
    should deposit MET and mint msdMET (depositFee == 0) (1408ms)
    should deposit TOKEN and mint msdTOKEN when TOKEN has transfer fee (1472ms)
    should deposit MET and mint msdMET (depositFee > 0) (1866ms)
    should deposit on behalf of another user (1418ms)
    should trigger rewards update (467ms)
  transfer
    should transfer if amount <= free amount (453ms)
    should revert if amount > free amount
    should add and remove deposit token from users' arrays only once (160ms)
    should trigger rewards update (426ms)
  transferFrom
    should transfer if amount <= free amount (488ms)
    should revert if amount > free amount (53ms)
    should trigger rewards update (503ms)
  seize
    should revert if not pool
    should seize tokens (101ms)
    should trigger rewards update (64ms)
  updateCollateralFactor
    should update collateral factor (49ms)
    should revert if using the current value
    should revert if not governor
    should revert if > 100%
  updateMaxTotalSupply
    should update max total supply (58ms)
    should revert if using the current value (99ms)
    should revert if not governor
  toggleIsActive
    should update min deposit time (93ms)
    should revert if not governor

E2E tests
synth mainnet end to end sanity tests
  should deposit (10971ms)
  should issue (11166ms)
  should increase debt by the time (2638ms)
  should liquidate unhealthy position (10113ms)
  should swap (8815ms)
  should repay (2330ms)
  should revert if repaying using wrong synthetic asset (2232ms)
  should withdraw (4987ms)

Integration tests
deposit
issue
swap
repay
  should revert if repaying using wrong synthetic asset (62ms)
withdraw
  should withdraw (555ms)
liquidate
  should liquidate unhealthy position (1244ms)

NativeTokenGateway
```



```
    should not receive ETH if sender is not WETH contract
deposit
    should deposit ETH to Pool (2895ms)
    should allow N deposits (1532ms)
withdraw
    should withdraw ETH from Pool (3240ms)

Pool
when user deposited multi-collateral
    should calculate deposit correctly
    should be able to issue using position among multiple collaterals (700ms)
when user deposited some MET
    should withdraw when collateral charges transfer fee (526ms)
when user minted some msETH
    liquidate
        should revert if amount to repay == 0
        should revert if liquidator == account
        should revert if position is healthy (46ms)
    when the position is unhealthy (collateral:debt >= 1)
        should not revert if paused (778ms)
        should revert if shutdown
        should revert if liquidator has not enough msAsset to repay (466ms)
        should revert if debt amount is < amount to repay
        should revert if repaying more than max allowed to liquidate (46ms)
        should liquidate by repaying all debt (protocolLiquidationFee == 0) (691ms)
        should liquidate by repaying all debt (protocolLiquidationFee > 0) (863ms)
        should liquidate by repaying > needed (protocolLiquidationFee == 0) (690ms)
        should liquidate by repaying > needed (protocolLiquidationFee > 0) (880ms)
        should liquidate by repaying < needed (protocolLiquidationFee == 0) (712ms)
        should liquidate by repaying < needed (protocolLiquidationFee > 0) (860ms)
        should liquidate by repaying the exact amount needed (protocolLiquidationFee == 0) (706ms)
        should liquidate by repaying the exact amount needed (protocolLiquidationFee > 0) (868ms)
    debt floor
        should revert if debt becomes < debt floor (50ms)
        should allow erase debt when debt floor set (813ms)
    when the position is unhealthy (collateral:debt < 1)
        should revert if paying more than needed to seize all deposit (39ms)
        should liquidate by repaying max possible amount (liquidateFee == 0) (658ms)
        should liquidate by repaying max possible amount (liquidateFee > 0) (1025ms)
        should liquidate by not repaying all debt (liquidateFee == 0) (728ms)
        should liquidate by not repaying all debt (liquidateFee > 0) (875ms)
    when user minted both msETH and msDOGE using all collateral
        should liquidate a position that have minted more than one msAsset (1369ms)
swap
    should not revert if paused (606ms)
    should revert if shutdown
    should revert if swap is paused
    should revert if amount == 0
    should revert if synthetic out is not active (45ms)
    should revert if user has not enough balance
    should swap synthetic tokens (swapFee == 0) (284ms)
    should swap synthetic tokens (swapFee > 0) (366ms)
Pause/Shutdown via PoolRegistry
    should pause pool if poolRegistry is paused
    should shutdown pool if poolRegistry is shutdown
whitelisting
addDebtToken
    should revert if not governor
    should add debt token
removeDebtToken
    should remove debt token (143ms)
    should revert if not governor
    should revert if debt token has any supply (44ms)
removeDepositToken
    should remove deposit token (148ms)
    should revert if not governor
    should revert if debt token has any supply (48ms)
updateTreasury
    should revert if using the same address
    should revert if caller is not governor
    should revert if address is zero
    should migrate funds to the new treasury (768ms)
updateSwapFee
    should revert if caller is not governor
    should revert if using the current value
    should revert if swap fee > 25%
    should update swap fee param
depositTokensOfAccount
addToDepositTokensOfAccount
    should revert if caller is not a deposit token
    should add deposit token to the account's array
    should revert when trying to add same deposit token twice (42ms)
removeFromDepositTokensOfAccount
    should revert if caller is not a deposit token
    should remove deposit token to the account's array (79ms)
debtTokensOfAccount
addToDebtTokensOfAccount
    should revert if caller is not a debt token
    should add debt token to the account's array
    should revert when trying to add same debt token twice (40ms)
removeFromDebtTokensOfAccount
    should revert if caller is not a debt token
    should remove debt token to the account's array (319ms)
updateDepositFee
    should revert if caller is not governor
    should revert if using the current value
    should revert if deposit fee > 25%
    should update deposit fee param
updateIssueFee
    should revert if caller is not governor
    should revert if using the current value
    should revert if issue fee > 25%
    should update issue fee param
updateWithdrawFee
    should revert if caller is not governor
    should revert if using the current value
    should revert if withdraw fee > 25%
    should update withdraw fee param
updateRepayFee
    should revert if caller is not governor
    should revert if using the current value
    should revert if repay fee > 25%
    should update repay fee param
updateLiquidatorIncentive
    should revert if caller is not governor
    should revert if using the current value
    should revert if liquidator incentive > 25%
    should update liquidator incentive param
updateProtocolLiquidationFee
    should revert if caller is not governor
    should revert if using the current value
    should revert if protocol liquidation fee > 25%
    should update protocol liquidation fee param
updateMaxLiquidable
    should revert if caller is not governor
    should revert if using the current value
    should revert if max liquidable > 100%
    should update max liquidable param
updateDebtFloor
    should revert if caller is not governor
    should revert if using the current value
    should update debt floor param
addRewardsDistributor
    should revert if caller is not governor
    should revert if null
    should revert if already added
    should add a rewards distributor
removeRewardsDistributor
    should revert if caller is not governor
    should revert if null
    should revert if not ealready added
    should remove a rewards distributor (134ms)
toggleIsSwapActive
    should toggle isSwapActive flag
    should revert if not governor

PoolRegistry
registerPool
    should revert if not governor
    should revert if pool is null
```

```

    should revert if adding twice
    should register pool (47ms)
    should manage pool ids (153ms)
unregisterPool
    should revert if not governor
    should revert if pool does not registered (79ms)
    should unregister pool (91ms)
updateFeeCollector
    should revert if not governor
    should revert if feeCollector is null
    should revert if using the same address
    should update fee collector
updateMasterOracle
    should revert if not governor
    should revert if using the same address
    should revert if address is zero
    should update master oracle contract (43ms)

RewardDistributor
updateTokenSpeed
    should revert if not governor
    should revert if not valid token
    should turn on (96ms)
    should update speed (224ms)
    should turn off (527ms)
updateTokenSpeeds
    should revert if not governor
    should update speeds (342ms)
supply actions
claimable
    should update rewards (from 0 to all supply)
    should update rewards (from 0 to half supply)
    should update rewards (from total to half supply) (113ms)
    should update rewards (from half to total supply) (111ms)
updateBeforeMintOrBurn
    should update rewards (from 0 to all supply) (106ms)
    should update rewards (from 0 to half supply) (116ms)
    should update rewards (from total to half supply) (188ms)
    should update rewards (from half to total supply) (193ms)
updateBeforeTransfer
    should update rewards on transfer (276ms)
claiming
    claimRewards (3800ms)
    claimRewards(address,address[]) (1229ms)
    claimRewards(address[],address[]) (1548ms)

SyntheticToken
default values (42ms)
mint
    should mint (122ms)
    should revert if not authorized
    should revert if surpass max supply in usd (97ms)
    should revert if msAsset is inactive (105ms)
burn
    should burn (436ms)
    should revert if not authorized
toggleIsActive
    should update active flag (102ms)
    should revert if not governor
updateMaxTotalSupply
    should update collateral factor (93ms)
    should revert if using the current value (40ms)
    should revert if not governor

Treasury
pull
    should revert if not deposit token
    should revert if amount == 0
    should pull MET tokens (772ms)

Pauseable
pause
    should revert if caller is not governor
    should revert if already paused (66ms)
    should pause (77ms)
unpause
    should revert if caller is not governor
    should revert if not paused (188ms)
    should revert if shutdown (61ms)
    should unpause (203ms)
open
    should revert if caller is not governor
    should revert if not shutdown (61ms)
    should open (99ms)
shutdown
    should revert if caller is not governor
    should revert if already shutdown (70ms)
    should shutdown (103ms)

TokenHolder
sweep
    should revert if caller is not sweeper
    should release token from contract (381ms)
    should sweep ETH (268ms)

309 passing (9m)
```

## Code Coverage

Overall code coverage is not very high, Quantstamp recommends adding additional tests to increase the coverage. We highly recommend increasing the branch coverage to at least 90% before going live.

| File                   | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|------------------------|---------|----------|---------|---------|-----------------|
| contracts/             | 94.17   | 78.13    | 95      | 94.84   |                 |
| DebtToken.sol          | 99.03   | 87       | 100     | 99.34   | 139             |
| DepositToken.sol       | 92.52   | 71.05    | 93.33   | 94      | ... 219,242,274 |
| NativeTokenGateway.sol | 100     | 75       | 100     | 100     |                 |
| Pool.sol               | 99.34   | 85.2     | 100     | 99.52   | 397             |
| PoolRegistry.sol       | 100     | 90       | 100     | 100     |                 |
| RewardsDistributor.sol | 97.3    | 67.86    | 100     | 94.79   | ... 260,261,284 |
| SyntheticToken.sol     | 62.75   | 58.62    | 72.73   | 70.67   | ... 201,203,204 |
| Treasury.sol           | 100     | 72.22    | 100     | 100     |                 |
| contracts/access/      | 75      | 53.85    | 76.92   | 79.17   |                 |
| Governable.sol         | 57.14   | 41.67    | 66.67   | 61.54   | 77,78,79,80,81  |



| File                           | % Stmts | % Branch | % Funcs | % Lines | Uncovered Lines |
|--------------------------------|---------|----------|---------|---------|-----------------|
| Manageable.sol                 | 100     | 64.29    | 85.71   | 100     |                 |
| contracts/interfaces/          | 100     | 100      | 100     | 100     |                 |
| IDebtToken.sol                 | 100     | 100      | 100     | 100     |                 |
| IDepositToken.sol              | 100     | 100      | 100     | 100     |                 |
| IGovernable.sol                | 100     | 100      | 100     | 100     |                 |
| IManageable.sol                | 100     | 100      | 100     | 100     |                 |
| INativeTokenGateway.sol        | 100     | 100      | 100     | 100     |                 |
| IPauseable.sol                 | 100     | 100      | 100     | 100     |                 |
| IPool.sol                      | 100     | 100      | 100     | 100     |                 |
| IPoolRegistry.sol              | 100     | 100      | 100     | 100     |                 |
| IRewardsDistributor.sol        | 100     | 100      | 100     | 100     |                 |
| ISyntheticToken.sol            | 100     | 100      | 100     | 100     |                 |
| ITreasury.sol                  | 100     | 100      | 100     | 100     |                 |
| contracts/interfaces/external/ | 100     | 100      | 100     | 100     |                 |
| ICToken.sol                    | 100     | 100      | 100     | 100     |                 |
| ICurveAddressProvider.sol      | 100     | 100      | 100     | 100     |                 |
| ICurveRegistry.sol             | 100     | 100      | 100     | 100     |                 |
| IMasterOracle.sol              | 100     | 100      | 100     | 100     |                 |
| IMulticall.sol                 | 100     | 100      | 100     | 100     |                 |
| IWETH.sol                      | 100     | 100      | 100     | 100     |                 |
| contracts/lib/                 | 93.1    | 80       | 92.86   | 94.12   |                 |
| MappedEnumerableSet.sol        | 92      | 83.33    | 91.67   | 93.33   | 72,121          |
| WadRayMath.sol                 | 100     | 75       | 100     | 100     |                 |
| contracts/mock/                | 100     | 100      | 100     | 100     |                 |
| PauseableMock.sol              | 100     | 100      | 100     | 100     |                 |
| contracts/storage/             | 100     | 100      | 100     | 100     |                 |
| DebtTokenStorage.sol           | 100     | 100      | 100     | 100     |                 |
| DepositTokenStorage.sol        | 100     | 100      | 100     | 100     |                 |
| PoolRegistryStorage.sol        | 100     | 100      | 100     | 100     |                 |
| PoolStorage.sol                | 100     | 100      | 100     | 100     |                 |
| RewardsDistributorStorage.sol  | 100     | 100      | 100     | 100     |                 |
| SyntheticTokenStorage.sol      | 100     | 100      | 100     | 100     |                 |
| TreasuryStorage.sol            | 100     | 100      | 100     | 100     |                 |
| contracts/upgraders/           | 73.91   | 33.33    | 83.33   | 88.46   |                 |
| DebtTokenUpgrader.sol          | 100     | 100      | 100     | 100     |                 |
| DepositTokenUpgrader.sol       | 100     | 100      | 100     | 100     |                 |
| PoolRegistryUpgrader.sol       | 100     | 100      | 100     | 100     |                 |
| PoolUpgrader.sol               | 100     | 100      | 100     | 100     |                 |
| RewardsDistributorUpgrader.sol | 0       | 100      | 0       | 0       | 9,14,15,16      |
| SyntheticTokenUpgrader.sol     | 100     | 100      | 100     | 100     |                 |
| TreasuryUpgrader.sol           | 100     | 100      | 100     | 100     |                 |
| UpgraderBase.sol               | 68.75   | 33.33    | 75      | 73.68   | 32,33,35,37,38  |
| contracts/utils/               | 100     | 96.67    | 92.31   | 92.31   |                 |
| Pauseable.sol                  | 100     | 96.43    | 100     | 100     |                 |
| TokenHolder.sol                | 100     | 100      | 50      | 66.67   | 20,27           |
| All files                      | 93.13   | 77.62    | 92.66   | 93.77   |                 |

# 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

|                                                                   |                                                                       |
|-------------------------------------------------------------------|-----------------------------------------------------------------------|
| 1c0131fc463ca4a25e53e0a126b4f1a7e1fd6354e68ed9192e42aeb6c4082315  | ./contracts/DebtToken.sol                                             |
| 47e7ee665d0f98d8f24bbd6de4a881ad0fa7118d78877fe253ee8dfd1e5f38d4  | ./contracts/DepositToken.sol                                          |
| c25cb63ba39c515917a86fe17dd0d342399679335aad55723b15c82a106f440a  | ./contracts/NativeTokenGateway.sol                                    |
| d7114e7cb084ef3d57b5e9f0189d9fc25f19f70a1b7cdd2f40d6445a1ffb49a6  | ./contracts/Pool.sol                                                  |
| a05bf1cae9115145b4b92b716a00b70baeaaeaf68cef09c5887e8db71879ae13  | ./contracts/PoolRegistry.sol                                          |
| df87c79a0d820eee854738a295cc1e1c72d1ba5c397deadf698b43429e81698f  | ./contracts/RewardsDistributor.sol                                    |
| d84af6926b4bf0a05b54f997c6e2abb038a9b3e436350673d9b4e1f77406a79b  | ./contracts/SyntheticToken.sol                                        |
| 473bb1764d8b1b462efbc9dc3083cb1c5aca6b6d4455d8332994c0002ef0343a  | ./contracts/Treasury.sol                                              |
| 5177ee9bb9d0efb885eade850c3f90e309d6666373d69f3b6a279a141da83ca2  | ./contracts/utils/Pauseable.sol                                       |
| 2818e239f6d538e7dc6a74bdd24afe93aedac12dbe7c6df0249fbfc740325ea6  | ./contracts/utils/TokenHolder.sol                                     |
| 50173539577caa935cee9fb2ac12acbd956a71017f1433932675803dac74ff88  | ./contracts/upgraders/DebtTokenUpgrader.sol                           |
| 05fcd167e9cfdc7b4a826340deb14b2c0f564506dd8e8c855e5cc567580713be  | ./contracts/upgraders/DepositTokenUpgrader.sol                        |
| 7e5a84cf54711afd016cc1944ea730355a44277e63aecc168f84b1858ec693d5  | ./contracts/upgraders/PoolRegistryUpgrader.sol                        |
| 7adbc b15ed043df6a4defd347e5c734e5f680fa0226285ac1cf0b387863c83d7 | ./contracts/upgraders/PoolUpgrader.sol                                |
| 107d31486a6913f4ec06795bc25a7ea5c09409e138cd89d1fb09f1629b4f805e  | ./contracts/upgraders/RewardsDistributorUpgrader.sol                  |
| 3755f1b9802a530c9ac6e6f29eb90a4b4e89af4849386d1bf45b9829691637ac  | ./contracts/upgraders/SyntheticTokenUpgrader.sol                      |
| 1d02ac0de9ae c7985c336d43a795fdbd00df7fd654e7d009f64e4161fc10df5  | ./contracts/upgraders/TreasuryUpgrader.sol                            |
| ddf5f560df0c695723a9b54e069fca9ac90ec5a40744d5eff014d01f140ad96c  | ./contracts/upgraders/UpgraderBase.sol                                |
| f33c30c0ed3f49ae33b4c84d53bd1929e930c32f5e397d29dca88298333694eb  | ./contracts/storage/DebtTokenStorage.sol                              |
| 52afc99a29b9423abe90e4eed01a60c58b567f3f88925d7045a580d85e6fffb1  | ./contracts/storage/DepositTokenStorage.sol                           |
| 691617ac ec25a55c72922c351518c70c09ff829d201382796c072e3c60745f69 | ./contracts/storage/PoolRegistryStorage.sol                           |
| e0143024dfc2d1f2b79730c0db88f6275538332fe707d8f068f5a93b584e2015  | ./contracts/storage/PoolStorage.sol                                   |
| 65910967dc75349f39ed78ae88f3238e7ad51d5c43e3789052633b93823d4a3d  | ./contracts/storage/RewardsDistributorStorage.sol                     |
| 5e296dee82e5ee2198689729a0e3c0e5fca769a91885a6e671cc21772a36cae3  | ./contracts/storage/SyntheticTokenStorage.sol                         |
| 5f80675ce325513c8577a3fd1716f174ad53dcc2c71e31edecdc3a0c69dda5a1  | ./contracts/storage/TreasuryStorage.sol                               |
| 7d8bf757b71b3b3aadd4c42bd842ffd88b0f3df4d470c2187afe76716e95b63b  | ./contracts/mock/ERC20Mock.sol                                        |
| fb6af6b975555a296f6df8f943f32b0071f9080a67f71ca785c1ec48aaa20ea6  | ./contracts/mock/MasterOracleMock.sol                                 |
| c831087317895bbecc2433ec1b2756ac553274dbc04e299bbfb63ed9d9cf59dd  | ./contracts/mock/PauseableMock.sol                                    |
| c5a0951c8fb11476249ec085008ea3e74d673cf35d40d633ec6d154b2b5fcd2e  | ./contracts/mock/PoolMock.sol                                         |
| e7e6b3f07776e91567454d3635182f1640826ca123c66a625185bb138a10fc44  | ./contracts/mock/TokenHolderMock.sol                                  |
| 629a9588c184a4aa686bb723908f96a8f2dc45cf6499e26877373c287ef3a640  | ./contracts/lib/MappedEnumerableSet.sol                               |
| 720d108fb0076bac21179b4c6a771b708a347e895f94d7af7f03fda2e2aaa43c  | ./contracts/lib/WadRayMath.sol                                        |
| 5695583503a8f36ec15777f2005ff8de0b1a7ad46df3e73da0187f38ac2f7469  | ./contracts/interfaces/IDebtToken.sol                                 |
| 87836d9af8eef41495913f853fe33217444c8e51e5867b6ac7852be7174fd01a  | ./contracts/interfaces/IDepositToken.sol                              |
| 8dfb1fe8dd4e15b744112d7eddf64cd869c70787e37d86eb8b9ae83815c99e4b  | ./contracts/interfaces/IGovernable.sol                                |
| a7652b2dd643245f4b4b1610addb6bed5a32bed11bac1fe2d79a6091fdb241df  | ./contracts/interfaces/IManageable.sol                                |
| 7ca80efefc233b331bedd36ce76131a95961d8dd4dfcf69db510459d2670026   | ./contracts/interfaces/INativeTokenGateway.sol                        |
| bc707de84d972ac7f6bd890629e1c11ab1b0a520bf6c099ae908bcec97b6efaf  | ./contracts/interfaces/IPauseable.sol                                 |
| b756bacd47839fef7bec915b51dfadc5eb9fdf9cecb92076dc3545bbca2f296c  | ./contracts/interfaces/IPool.sol                                      |
| ded4ba98b69a5805bd3234d62da8a6c8a0a4a770da2fa6ada0112b6279fbf8e2  | ./contracts/interfaces/IPoolRegistry.sol                              |
| 96cd590d391417309830923fbd2885d58482e7c62d4f6ef35d09a94f62949a5b  | ./contracts/interfaces/IRewardsDistributor.sol                        |
| 896edc23c0919bc4381409dd4b3d5ab0ad12e76485ec7bc5c57e1d0ff71e67f7  | ./contracts/interfaces/ISyntheticToken.sol                            |
| 8564808395e5ef8a1dc5dd6f2b80381cc138a377e3a5a1e84274f4b53dfe58df  | ./contracts/interfaces/ITreasury.sol                                  |
| b2ba9faafa3fbc2b4c1fc37e457906dd53a56fe1b77b704746b4f2439b24b1f6  | ./contracts/interfaces/external/ICToken.sol                           |
| 5b11f5d97f6e766f572bf1a4e611f597ce380ff3358aa64ecfbc64766f90f8e06 | ./contracts/interfaces/external/ICurveAddressProvider.sol             |
| fa8285d12e197c9ae063b4209408309d7bcd2cafe91f32e76d4ed951a20abeff  | ./contracts/interfaces/external/ICurveRegistry.sol                    |
| 8f8081381c70a2a7b40ec0ffed9c6db10ce37faee2c9e74ff8a3fc1c56661d2e  | ./contracts/interfaces/external/IMasterOracle.sol                     |
| ead92c756b454cb85ed44bac55f95c62abcf1276738e47e9964668c10b804161  | ./contracts/interfaces/external/IMulticall.sol                        |
| ff3390196539d57f961b0d2705eb215740caa89f6650c17932a87c606c8a0902  | ./contracts/interfaces/external/IWETH.sol                             |
| 1370d859f5c6d11025afb409d1b724279f663c4cf4bc4d2ba057290bdcf45a66  | ./contracts/dependencies/openzeppelin/utils/Address.sol               |
| 5828bf38f9376b659a8edbbe2df0d06b29a09e37ecd470465dda2bbcb612c85d  | ./contracts/dependencies/openzeppelin/utils/Context.sol               |
| 40020f75929aa61b29ad51505de865754bac2d0939050b61ae076255609783af  | ./contracts/dependencies/openzeppelin/utils/StorageSlot.sol           |
| 17b9dd0046758767e35f41abe264bdb1893377cb666fb0ed176d3cd15acc7c38  | ./contracts/dependencies/openzeppelin/utils/structs/EnumerableSet.sol |



b1417f64b6fba54e2b715f2228c4dde4065e742245e6bf7c68f39c5f42af043b ./contracts/dependencies/openzeppelin/utils/math/Math.sol

8dec6dd63908459be4b909861eb3a00b31431575f8169c1f0cbb6519af754bfe ./contracts/dependencies/openzeppelin/utils/math/SafeCast.sol

80e33e340442acecc4bd995b4ead9b51adc4231c8213357fca18996b945f850b ./contracts/dependencies/openzeppelin/token/ERC20/ERC20.sol

b2565dec975f684ef0edfa505e212d0d0b602e1311afab782ea06ea8d3f49bb6 ./contracts/dependencies/openzeppelin/token/ERC20/IERC20.sol

729097c056b8bf1dd93ac16831380ce4ff54703d75983f57354240cc8be2edec ./contracts/dependencies/openzeppelin/token/ERC20/utils/SafeERC20.sol

4e2ce556a0419415ec3b01a0fa0322c20d6d53de5a05728c068e90d5684486c1  
./contracts/dependencies/openzeppelin/token/ERC20/extensions/IERC20Metadata.sol

990bec6996afd6ab53aa9509e1df1b115c7517982edc20c278b802c75ed91a4e ./contracts/dependencies/openzeppelin/security/ReentrancyGuard.sol

370b3075d01ef48b190f36120a06525df8428cc4684f1ae810381737f0d934b2 ./contracts/dependencies/openzeppelin/proxy/Proxy.sol

8954474eb8ba4a536daa0a690f24680587079d364172d9f965f367d2a9f25839 ./contracts/dependencies/openzeppelin/proxy/utils/Initializable.sol

199d09f80f7c835689f4ab95b9a0b60da7f57953ae5b9fb94f72dccb18b8a990 ./contracts/dependencies/openzeppelin/proxy/transparent/ProxyAdmin.sol

42f11913b19098e92becf0dd1ceec5e768657b76226777f33cfb8874667dfdb61  
./contracts/dependencies/openzeppelin/proxy/transparent/TransparentUpgradeableProxy.sol

13d890d68e3dba5ffa21db23a4a1cf77e691e3325dea87dd0c77e25e4fb27a85 ./contracts/dependencies/openzeppelin/proxy/ERC1967/ERC1967Proxy.sol

7c93848750383f042cbb89d48cb31a983ab3b4c52916f179697adf017f3989e8 ./contracts/dependencies/openzeppelin/proxy/ERC1967/ERC1967Upgrade.sol

312cb1d7e14511ac958fe4963fb2e4154f42d9939f005d5703def1a1a1f21aa5 ./contracts/dependencies/openzeppelin/proxy/beamon/IBeacon.sol

0195650aabf5270babe540969c56f8f244342aebce89266787a3b015e41d608f ./contracts/dependencies/openzeppelin/access/Ownable.sol

dc8833d0b81a39a3c5842e1c033e40cabee28cf168f883d0e416c2f6f8444830 ./contracts/access/Governable.sol

176fa1bcc43c86f6e91026166d9274b78c3e25a56add9a5491aa0730661a23b1 ./contracts/access/Manageable.sol

Tests

af95fdb75885c78cbf087d30db938bc822aefdcfbfaaa35f27fad09da8bb0af74 ./test/DebtToken.test.ts

2a5c59a1bc3d726f9926929dbc637aa6990ff3d6aa9c4f4767d17e55518587b7 ./test/Deployments.test.ts

1806d4db3d7ad2455d5ac585b939bb913431fe2cf96144d71451a196c8edfd25 ./test/DepositToken.test.ts

d504216a70df637816420a31ffd32a0d0e2c8e0bfc96dd4a68eed6693c926d3a ./test/E2E.test.ts

c7b37981c9fcce5c5c2fc004081b26b5e469f1e1e37f5a961deff85fa9ec8176 ./test/Integration.test.ts

f49c62520ae8f5fdac9aaed3d7002ae4591b021dba544de688d8f6b0dd3cb7c4 ./test/NativeTokenGateway.test.ts

95cd8100a0337ba3468941c251377d3b2508c52b086700c7a97af2b59df62e4b ./test/Pool.test.ts

5c62def10be3dfb576aa79a109b56e5dad8ee9dd3bd5939a81db32049bf7b72a ./test/PoolRegistry.test.ts

0d1b821b6cb31a4a9b78331f6c1d64784281f6b202c64fe76e4bfcd68e66a7ac ./test/RewardDistributor.test.ts

9f1e3a5397e8300e15d83948fcbb4c6fb0fa670b3a13c0cd4ed1b05efd2541ca ./test/SyntheticToken.test.ts

e579fccbf4178d66b08530821736d132ca9bd33e3c543268578e5a93fd3bb39a ./test/Treasury.test.ts

84bbe810f6cc50880c15da1169d8b5b062b2feeece3ccf53e7eefab622479d5 ./test/utils/Pauseable.test.ts

b910bec16a68bd4272c4169017f9e999d7e992dd23d7a6c0e9bd2a748d9e1be0 ./test/utils/TokenHolder.test.ts

9c6bcd64e78d9dc5ce858a1abc40332e5ccdd6d2a726675f11f49a45cff1b668 ./test/helpers/index.ts

Changelog

- 2022-11-29 - Initial report
- 2022-12-15 - Final report

# About Quantstamp

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

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

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

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

## Notable Collaborations & Customers:

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

## Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

## Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

## Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

## Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

