Quantstamp Contract Security Certificate

December 6th 2019 — Quantstamp Verified

Pool Together Token

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Token
Auditors	Alex Murashkin, Senior Software Engineer Martin Derka, Senior Research Engineer Kacper Bąk, Senior Research Engineer
Timeline	2019-11-06 through 2019-12-05
EVM	Byzantium
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Repository README

Source Code

Repository	Commit
pooltogether-contracts	33e54bd

Changelog

- 2019-11-18 - Initial report (`3b4a607`)
- 2019-11-22 - Audited diff (`3b4a607..b59fd63`)
- 2019-11-29 - Audited diff (`b59fd63..78ac686`)
- 2019-12-05 - Final report (`33e54bd`)

Overall Assessment

The code is well-written, well-tested, and mostly well-documented. We have not found any significant security vulnerabilities, but identified 6 findings. We classified two of them as low-risk since they are unlikely to occur and have low impact. Two were deemed informational-level, and the remaining two were marked as "undetermined" due to lack of the necessary information. Four of the findings have been addressed in the code as of the commit `78ac686`.

It is important to note the following:

1. The code assumes that the used Compound token contract is a well-behaved, trustworthy ERC20 token.
2. The current approach to randomization, as noted in the previous audits, has shortcomings: e.g., the outcome of the lottery can be impacted by malicious admins or transaction-ordering. The team is aware of these, and we were informed that improving randomization is a future work.

Total Issues		(4 Fixed)
High Risk Issues	0	(0 Fixed)
Medium Risk Issues	0	(0 Fixed)
Low Risk Issues	2	(1 Fixed)
Informational Risk Issues	2	(1 Fixed)
Undetermined Risk Issues	2	(2 Fixed)

0 Unfixed
2 Acknowledged
4 Fixed

High	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	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	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.

Unfixed	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.

Summary of Findings

ID	Description	Severity	Status
QSP-1	Denial-of-Service (DoS)	⌄ Low	Resolved
QSP-2	Allowance Double-Spend Exploit	⌄ Low	Acknowledged
QSP-3	Centralization of Power	○ Informational	Acknowledged
QSP-4	Potential State of "Anarchy"	○ Informational	Resolved
QSP-5	Undefined Behaviour	? Undetermined	Resolved
QSP-6	Potential Issues in Reset Logic	? Undetermined	Resolved

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:

- Truffle
- Ganache
- SolidityCoverage
- Mythril
- Slither

Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed Ganache: `npm install -g ganache-cli`
3. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
4. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
5. Installed the Mythril tool from Pypi: `pip3 install mythril`
6. Ran the Mythril tool on each contract: `myth -x path/to/contract`
7. Installed the Slither tool: `pip install slither-analyzer`
8. Run Slither from the project directory `slither .`

Assessment

Findings

QSP-1 Denial-of-Service (DoS)

Severity: Low

Status: Resolved

File(s) affected: `BasePool.sol`

Description: A Denial-of-Service (DoS) is a situation which a smart contract can become unusable.
In `BasePool.sol`, `L228` (commit `3b4a607`) is possible to call `open(bytes32 _secretHash)` with an invalid (incorrect) hash. When this happens, an admin can no longer find a secret and a salt such that the line 290 in `BasePool.sol` does not revert. Therefore, the pool can no longer be rewarded.

Recommendation: While the function is `onlyAdmin` and admins are trusted, it is still recommended to consider ways for mitigating this potential issue. A potential solution may include an emergency recover function that can bypass the mistakenly provided secret.

QSP-2 Allowance Double-Spend Exploit

Severity: Low

Status: Acknowledged

File(s) affected: `Pool.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens. An example of an exploit goes as follows:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` 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 `approve()` 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 `transferFrom()` 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 an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens. The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance` and `decreaseAllowance`.

Recommendation: Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `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 contract.

QSP-3 Centralization of Power

Severity: Informational

Status: Acknowledged

File(s) affected: `BasePool.sol`

Description: In `BasePool.sol`, `L286` (commit `3b4a607`), the logic of the function `reward(lastSecret, _salt)` appears to be centralized. The random number based on the entropy is calculated within the context of a block, and a malicious admin can choose to call this function when it is beneficial for them.

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 admins.

QSP-4 Potential State of "Anarchy"

Severity: *Informational*

Status: Resolved

Description: `BasePool.sol`, `L614` (commit `3b4a607`): `removeAdmin(...)` allows the last admin to remove themselves. If that happens, the contract ends up having no admin.

Recommendation: Restricting the number of admins to be at least one by not allowing removal of an admin unless there is an additional admin left.

QSP-5 Undefined Behaviour

Severity: *Undetermined*

Status: Resolved

File(s) affected: `BasePool.sol`

Description: `BasePool.sol`, `L242`(commit `3b4a607`): While the method name suggests a state-changing action, the `commit()` method only emits an event and does not do any state change. The expected behaviour is not explicitly clarified.

Recommendation: To clarify the expected behaviour or rename the method to better reflect the actual behaviour.

QSP-6 Potential Issues in Reset Logic

Severity: *Undetermined*

Status: Resolved

File(s) affected: `DrawManager.sol`

Description: In `DrawManager.sol`, `L207` and `L212` (commit `3b4a607`), the draw index is being deleted while the corresponding draw sets are not being reset. This is inconsistent with the logic of the `withdrawCommitted` method that, upon deletion, also resets the draw sets.

Recommendation: Considering adding the following lines of code:

- In `DrawManager.sol`, `L207`, the second `drawSet` should likely be reset by adding the line `drawSet(state, secondDrawIndex, 0, user);`
- In `DrawManager.sol`, `L212`, the first `drawSet` should likely be reset by adding the line: `drawSet(state, firstDrawIndex, 0, user);`

Automated Analyses

Mythril

Mythril reported several instances of integer overflow and exception states which are deemed to be false-positives. In addition, calls to the Compound contract were flagged, however, these were also deemed to be false-positives under the assumption that the Compound contract is referenced correctly and trusted. There were also multiple instances of "The contract account state is changed after an external call" which were deemed to be not having security implications.

Slither

Slither reported six potential re-entrancies which are deemed to be false-positives. In addition, Slither suggested to initialize the local variables in the `withdrawCommited(...)` method, which was included in the Best Practices recommendations (and already addressed). There were other reported info-level findings which were deemed to be not having security implications.

Code Documentation

1. Most methods are well-documented, however, some public or external methods are still missing documentation. We recommend documenting all the methods as it help to clarify the expected behaviour. For example, methods `depositCommitted()` and `withdrawCommitted()` in `DrawManager.sol` (commit `3b4a607`) are not documented - **Fixed**
2. `ERC777Pool.sol`, `L514`: `requireReceptionAck` (commit `78ac686`) is not documented - **Fixed**
3. `BasePool.sol.L306`: `rewardAndOpenNextDraw()` and `L321`: `reward()` (commit `78ac686`) missing descriptions of the `_salt` parameter - **Fixed**
4. `MCDAwarePool.sol`, `L77` (commit `78ac686`): a typo in "initialize" - **Fixed**

Adherence to Best Practices

`DrawManager.sol`, `L210` (commit `3b4a607`): a letter in the comment appears to be missing: `needs to be destroye` -> `needs to be destroyed`- **Fixed**

`BasePool.sol`, `L56` (commit `3b4a607`): the value expression in `ETHER_IN_WEI = 1000000000000000000` is written using 18 consecutive zeros. A better practice from the readability standpoint would be rewriting it as `10 ** 18` - **Fixed**

`DrawManager.sol`, `L184-186` (commit `3b4a607`): the local variables should be initialized to `0` before use - **Fixed**

`DrawManager.sol`, `L287` (commit `3b4a607`): a good practice would be asserting the input address `_addr` is non-zero - **Fixed**

Test Results

Test Suite Results

All tests are passing. The stress test, when enabled, also passes.

```
  Contract: BasePool
    init()
       ✓ should fail if owner is zero (6375972 gas)
       ✓ should fail if moneymarket is zero (6375812 gas)
    addAdmin()
       ✓ should allow an admin to add another (45336 gas)
       ✓ should not allow a non-admin to remove an admin (23498 gas)
    removeAdmin()
       ✓ should allow an admin to remove another (15823 gas)
       ✓ should not allow a non-admin to remove an admin (23588 gas)
       ✓ should not an admin to remove an non-admin (23953 gas)
       ✓ should not allow an admin to remove themselves (23996 gas)
    supplyRatePerBlock()
       ✓ should work (6866414 gas)
    committedBalanceOf()
       ✓ should return the users balance for the current draw (7452455 gas)
    openBalanceOf()
       ✓ should return the users balance for the current draw (7452583 gas)
    estimatedInterestRate()
       ✓ should set an appropriate limit based on max integers (6866414 gas)
    getDraw()
       ✓ should return empty values if no draw exists (6866414 gas)
       ✓ should return true values if a draw exists (7100888 gas)
    openNextDraw()
       ✓ should have opened a draw
       ✓ should emit a committed event (189090 gas)
       ✓ should revert when the committed draw has not been rewarded (214701 gas)
       ✓ should succeed when the committed draw has been rewarded (493197 gas)
    reward()
       ✓ should fail if there is no committed draw (26777 gas)
       ✓ should fail if the committed draw has already been rewarded (330910 gas)
       ✓ should fail if the secret does not match (236316 gas)
       ✓ should award the interest to the winner (877719 gas)
       ✓ can only be run by an admin (215595 gas)
    rolloverAndOpenNextDraw()
       ✓ should not run if there is no committed draw (24948 gas)
       ✓ should not run if the committed draw has already been rewarded (329081 gas)
       ✓ should only be run by an admin (213360 gas)
       ✓ should rollover the draw and open the next (405502 gas)
    rollover()
       ✓ should only be called by admin (211076 gas)
       ✓ should not run if there is no committed draw (22258 gas)
       ✓ should not run if the committed draw has been rewarded (326391 gas)
       ✓ should reward the pool with 0 (426531 gas)
    rewardAndOpenNextDraw()
       ✓ should revert if there is no committed draw (29271 gas)
       ✓ should fail if the secret does not match (240602 gas)
    depositPool()
       ✓ should fail if there is no open draw (67971 gas)
    with a fresh pool
      transfer()
         ✓ should transfer tickets to another user (909456 gas)
      depositPool()
         ✓ should fail if not enough tokens approved (120865 gas)
         ✓ should deposit some tokens into the pool (310165 gas)
         ✓ should allow multiple deposits (467267 gas)
      depositSponsorship()
         ✓ should contribute to the winnings (961115 gas)
      withdraw()
         ✓ should work for one participant (1293128 gas)
         ✓ should work for two participants (1627052 gas)
         ✓ should work when one user withdraws before the next draw (1460585 gas)
        with sponsorship
           ✓ should allow the sponsor to withdraw partially (48790 gas)
      balanceOf()
         ✓ should return the entrants total to withdraw (310165 gas)
    when fee fraction is greater than zero
       ✓ should reward the owner the fee (8251850 gas)
    when a pool is rewarded without a winner
       ✓ should save the winnings for the next draw (8268284 gas)
    setNextFeeFraction()
       ✓ should allow the owner to set the next fee fraction (43557 gas)
       ✓ should not allow anyone else to set the fee fraction (22565 gas)
       ✓ should require the fee fraction to be less than or equal to 1 (66261 gas)
    setNextFeeBeneficiary()
       ✓ should allow the owner to set the next fee fraction (29925 gas)
       ✓ should not allow anyone else to set the fee fraction (23587 gas)
       ✓ should not allow the beneficiary to be zero (22327 gas)
    pause()
       ✓ should not allow any more deposits (111670 gas)
    unpause()
       ✓ should not work unless paused (21946 gas)
       ✓ should allow deposit after unpausing (367936 gas)
    transferBalanceToSponsorship()
       ✓ should transfer the balance of the pool in as sponsorship (162109 gas)

  Contract: DrawManager
    openNextDraw()
       ✓ should create a draw when none is available (156384 gas)
      when there is an existing draw
         ✓ should create the next draw (86943 gas)
  openSupply()
     ✓ should return 0 if no draw exists
  deposit()
     ✓ should fail if there is no current draw (25784 gas)
    when a draw has been opened
       ✓ should fail if the address is zero (24542 gas)
       ✓ should deposit the tokens as open tokens (179465 gas)
      when the user has already deposited
         ✓ should allow them to deposit again (43201 gas)
        and a second draw has been opened
           ✓ should make the previous balance eligibile and start a new open balance (179543 gas)
          and the user has deposited, and there is a third open
             ✓ should collapse the previous two draws and update the open draw (175394 gas)
  openBalanceOf()
     ✓ should return 0 when no draw exists
    when an open draw exists
       ✓ should return the open balance of the user
    when an open draw has passed
       ✓ should reflect the current open draw only
  committedBalanceOf()
     ✓ should return 0 when no draw exists
    when a committed draw exists
       ✓ should return the committed balance of the user
      and the user has deposited multiple times
         ✓ should return the total of both draws
  balanceOf
     ✓ should return 0 if nothing exists
  depositCommitted()
     ✓ should fail if the address is zero (24542 gas)
     ✓ should work when recipient already has committed deposits (366613 gas)
     ✓ should fail when there is no committed draw (6617374 gas)
     ✓ should work when recipient has no committed deposits (451594 gas)
  withdrawCommitted()
     ✓ should fail if the address is zero (24563 gas)
     ✓ should allow a user to withdraw their committed tokens (254466 gas)
     ✓ should allow a user to withdraw their committed tokens when they also have open tokens (390879 gas)
     ✓ should allow a user to withdraw partial committed tokens when they have two committed draws (623694 gas)
     ✓ should allow a user to fully withdraw committed tokens when they have two committed draws (609614 gas)
     ✓ should not withdraw open tokens (26786 gas)
  withdraw()
     ✓ should fail if the address is zero (23958 gas)
     ✓ should allow the user to withdraw their open tokens (47952 gas)
    when both open and eligible balances
       ✓ should allow the user to withdraw all of their tokens (114512 gas)
  draw
     ✓ should return address(0) if no eligible deposits
    with open deposits
       ✓ should return 0
      and they become eligible
         ✓ should work
        drawWithEntropy()
           ✓ should work
        and one withdraws
           ✓ should fail with the previous total
           ✓ should read the original depositers
        and there is a second round of deposits
           ✓ should draw from them all
           ✓ should fail with an invalid token
  drawWithEntropy()
     ✓ should return the 0 address if no entries

Contract: ERC777Pool
  initERC777()
     ✓ requires the name to be defined (25609 gas)
     ✓ requires the symbol to be defined (25757 gas)
  with a pool with a default operator
    initERC777()
       ✓ should add the default operators
       ✓ cannot be called twice (26499 gas)
    revokeOperator()
       ✓ should allow users to revoke the operator (45336 gas)
    authorizeOperator()
       ✓ should allow users to revoke the default operator then add them back (60668 gas)
    operatorBurn()
       ✓ should not allow someone to burn the zero address tokens (24657 gas)
    operatorSend()
       ✓ should not send tokens from zero address (25945 gas)
  with a fully initialized pool
    initERC777()
       ✓ should setup the name, symbol and register itself with ERC 1820
    decimals()
       ✓ should equal 18
    granularity()
       ✓ the smallest indivisible unit should be 1
    totalSupply()
       ✓ total supply should be correct (897243 gas)
    send()
       ✓ should send tokens to another user (794632 gas)
       ✓ should revert if sending to the burner address (609331 gas)
       ✓ should work if sending zero (69453 gas)
      when sender has IERC777Sender interface
         ✓ should call the interface (907839 gas)
      when recipient has IERC777Recipient interface
         ✓ should call the interface (947025 gas)
      when recipient does not have IERC777Recipient interface
         ✓ should succeed for EOA addresses (872025 gas)
         ✓ should fail for contract addresses without ERC777Recipient interfaces (1460383 gas)
    transfer()
       ✓ should transfer tokens to another user (871637 gas)
       ✓ should revert if transferring to the burner address (608708 gas)
       ✓ should work if transferring zero (118060 gas)
       ✓ should reject when transferring to zero address (22091 gas)
    burn()
```

```
         ✓ should be okay to burn nothing (119634 gas)
         ✓ should allow a user to burn some of their tokens (733723 gas)
      isOperatorFor()
         ✓ should be that a user is an operator for themselves
         ✓ should be false when a non-operator is checked
         ✓ should be true when someone is added as an operator (45328 gas)
      authorizeOperator()
         ✓ should allow someone to add an operator (45328 gas)
         ✓ should not allow someone to add themselves (23212 gas)
      revokeOperator()
         ✓ should allow someone to revoke an operator (60648 gas)
         ✓ should not allow someone to revoke themselves (23210 gas)
      defaultOperators()
         ✓ should be an empty array
      operatorSend()
         ✓ should allow an operator to send tokens on behalf of another user (950213 gas)
         ✓ should not allow an non-authorized operator to send tokens on behalf of another user (613100 gas)
         ✓ should not allow an operator to send from the zero address (25587 gas)
         ✓ should not allow an operator to send to the zero address (657148 gas)
         ✓ should not allow an operator to send more tokens than their balance (658428 gas)
      operatorBurn()
         ✓ should allow an operator to burn someones tokens (772981 gas)
         ✓ should not allow an non-authorized operator to burn someones tokens (611815 gas)
         ✓ should not allow someone to burn the zero address tokens (24302 gas)
         ✓ should not allow the burn to exceed the balance (661445 gas)
      allowance() & approve()
         ✓ should not allow someone to approve the zero address (22459 gas)
         ✓ should return the number of tokens that are approved to spend (45690 gas)
      transferFrom()
         ✓ should allow a spender to transfer tokens (345102 gas)
         ✓ should fail if a spender tries to spend more than their allowance (500752 gas)
         ✓ should fail if the recipient is zero (69491 gas)
         ✓ should fail if the from is zero (69526 gas)

  Contract: MCDAwarePool
    tokensReceived()
      from ERC777
         ✓ should fail (138333 gas)
      from an MCDAwarePool
         ✓ should migrate the sai to dai and deposit (521260 gas)
        when the dai pool has a committed draw
           ✓ should migrate the sai to dai and immediately have a balance (613405 gas)
        to a non-Dai MCD Pool
           ✓ should reject the transfer (425716 gas)
    initBasePoolUpgrade()
       ✓ should safely upgrae the pool (7988631 gas)

  Contract: Pool
    scdMcdMigration()
       ✓ should return the right address
    saiPool()
       ✓ should return the correct address

  Contract: ERC777Pool
    with a fully initialized pool
      setRecipientWhitelistEnabled(bool _enabled)
         ✓ should work (42441 gas)
      recipientWhitelistEnabled()
         ✓ should enable the whitelist (42441 gas)
         ✓ should disable the whitelist (56130 gas)
      setRecipientWhitelisted(address _recipient, bool _whitelisted)
         ✓ should work (43953 gas)
      recipientWhitelisted(address _recipient)
         ✓ should test whether an address is whitelisted (43953 gas)
      with whitelisting enabled
        transfer()
           ✓ should not be allowed (24418 gas)
           ✓ should be allowed for whitelist (252067 gas)
        send()
           ✓ should not be allowed (24986 gas)
           ✓ should be allowed for whitelist (252528 gas)
        burn()
           ✓ should work (120115 gas)

  Contract: StressTest
    - should work

  Contract: ExposedUniformRandomNumber
    uniform()
       ✓ should return 0 if the upper bound is zero
       ✓ should skip the first X numbers that cause modulo bias

  161 passing (5m)
  1 pending



  Contract: StressTest
Draw 0: Deposit 0: 136264
Draw 0: Deposit 1: 121506
Draw 0: Deposit 2: 121506
Draw 0: Deposit 3: 121506
Draw 0: Deposit 4: 121506
Draw 0: Deposit 5: 121506
Draw 0: Deposit 6: 121506
Draw 0: Deposit 7: 121506
Draw 0: Deposit 8: 121506
Draw 0: Deposit 9: 121506
Draw 0: Deposit 10: 169144
Draw 0: Deposit 11: 127587
Draw 0: Deposit 12: 127587
Draw 0: Deposit 13: 127587
Draw 0: Deposit 14: 127587
Draw 0: Deposit 15: 127587
Draw 0: Deposit 16: 127587
Draw 0: Deposit 17: 127587
Draw 0: Deposit 18: 127587
Draw 0: Deposit 19: 169080
Draw 0: Withdraw 19: 50966
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 0: Withdraw 19: 25682
Draw 1: Deposit 0: 136303
Draw 1: Deposit 1: 121545
Draw 1: Deposit 2: 121545
Draw 1: Deposit 3: 121545
Draw 1: Deposit 4: 121545
Draw 1: Deposit 5: 121545
Draw 1: Deposit 6: 121545
Draw 1: Deposit 7: 121545
Draw 1: Deposit 8: 121545
Draw 1: Deposit 9: 121545
Draw 1: Deposit 10: 169183
Draw 1: Deposit 11: 127626
Draw 1: Deposit 12: 127626
Draw 1: Deposit 13: 127626
Draw 1: Deposit 14: 127626
Draw 1: Deposit 15: 127626
Draw 1: Deposit 16: 127626
Draw 1: Deposit 17: 127626
Draw 1: Deposit 18: 127626
Draw 1: Deposit 19: 169080
Draw 1: Withdraw 19: 50966
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 1: Withdraw 19: 25682
Draw 2: Deposit 0: 183135
Draw 2: Deposit 1: 153377
Draw 2: Deposit 2: 156205
Draw 2: Deposit 3: 156205
Draw 2: Deposit 4: 156205
Draw 2: Deposit 5: 156205
Draw 2: Deposit 6: 156205
Draw 2: Deposit 7: 156205
Draw 2: Deposit 8: 156205
Draw 2: Deposit 9: 156205
Draw 2: Deposit 10: 216015
Draw 2: Deposit 11: 174458
Draw 2: Deposit 12: 174458
Draw 2: Deposit 13: 174458
Draw 2: Deposit 14: 174458
Draw 2: Deposit 15: 174458
Draw 2: Deposit 16: 174458
Draw 2: Deposit 17: 174458
Draw 2: Deposit 18: 148889
Draw 2: Deposit 19: 169080
Draw 2: Withdraw 19: 50966
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
Draw 2: Withdraw 19: 25682
     ✓ should work (10625799 gas)
```

Code Coverage

The code has full coverage in terms of statements, lines, and functions. While achieving the 100% branch coverage may not be necessary, we recommend paying attention to it as there could be unseen edge-case scenarios when the smart contract has unexpected behaviour.

The method `_callTokensToSend` (`ERC777Pool.sol`, `L491-493`, commit `78ac686`) that was not covered in commit `78ac686`, is covered as of commit `33e54bd`.

```
-------------------------------------|----------|----------|----------|----------|----------------|
File                                 |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
-------------------------------------|----------|----------|----------|----------|----------------|
 contracts/                          |      100 |     96.1 |      100 |      100 |                |
    BasePool.sol                     |      100 |       88 |      100 |      100 |                |
    DrawManager.sol                  |      100 |      100 |      100 |      100 |                |
    ERC777Pool.sol                   |      100 |      100 |      100 |      100 |                |
    MCDAwarePool.sol                 |      100 |      100 |      100 |      100 |                |
    Pool.sol                         |      100 |      100 |      100 |      100 |                |
    RecipientWhitelistERC777Pool.sol |      100 |      100 |      100 |      100 |                |
    UniformRandomNumber.sol          |      100 |      100 |      100 |      100 |                |
-------------------------------------|----------|----------|----------|----------|----------------|
All files                            |      100 |     96.1 |      100 |      100 |                |
-------------------------------------|----------|----------|----------|----------|----------------|
```

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
6842bb7d1ead78b72b760b449ab6f25819591072346bb980ed4c00f638bca015 ./contracts/BasePool.sol
9088148ee04306206fbf78bafe6c27fe7902fc4eff4b2fcd3338838cc0f4a96e ./contracts/DrawManager.sol
c42a441c48138e9970b41ec6e1b35539d5be4067d5dbeaee5f3451e841e7ef54 ./contracts/ERC777Pool.sol
005f4aa8e6f821bf820d0413907c2687532047d3ba4c367722b9ed894baac14f ./contracts/MCDAwarePool.sol
7910ff3916e09c5a190ddd1d7c0a13027d12561ffaec88a99c5526cf287bbeb4 ./contracts/Migrations.sol
66c177d18a78de69cf837be83923a82e7c3ee53d16a666e45968e905123b14fe ./contracts/Pool.sol
908f63bee0db41f9537cc6946afd2e4f616b6cb067c723a2523e3166e05d67dc ./contracts/RecipientWhitelistERC777Pool.sol
431d88c926d0aa689b22f5ee3f38c029b632ca09fd2e04599bfb39e2c8ea3b5a ./contracts/UniformRandomNumber.sol
eb3204f5fceb0d9984293db3853b7125098dfc1764758cc42c950da143e21880 ./contracts/test/CErc20Mock.sol
9761621bae4066561cddeefb2e20b36ec22cf9dd3806d35c6e61c7f9be4b2930 ./contracts/test/ERC777Mintable.sol
93c430b23267c01961bbec7dd174116ae09142cdb286af513d5139fcd9f9aa19 ./contracts/test/ExposedDrawManager.sol
a169ba98f2dce5d54bee8960805397d9804b6c636ba6eba5a971c3f9a9126bed ./contracts/test/ExposedUniformRandomNumber.sol
daa2340d6864b3a28348dc2d8444086cb201b01a14b476aaa0116d35abbcf34d ./contracts/test/LocalMCDAwarePool.sol
aa91f729c6c728ea15095c51e54baaf62e3b869d904ff86cfbb29fb40785f1d7 ./contracts/test/MockERC777Recipient.sol
105a321ba481ec01582150fa8f5d9cdf4b656902bb16671ec9b13c7e4c6394a9 ./contracts/test/MockERC777Sender.sol
106ee3e31ded379d86f0a654aadccc4e04c9ae865e34b91ee5989351d53112f5 ./contracts/test/Token.sol
915ce9901088c08535c96d90ff4f592d8438b957a0a2e8dab184d66310790488 ./contracts/test/maker/MockJoinLike.sol
5593bb41ca67b5a39fc88de5ae82b368e1ba20d3289a40c3376663a340cafd88 ./contracts/test/maker/MockScdMcdMigration.sol
31d0360a63b7982003ae4b4da8959877617b38cc0e91318388412ced4cea638b ./contracts/compound/ICErc20.sol
```

Tests

```
097b4b94402c016c1508522de30538e7c43a00d010783405d1b479d9669a9454 ./test/BasePool.test.js
a5f322e475b22ebe14fe5741e0dff6629e14f573a43d74a718d2681b8aa75c9c ./test/DrawManager.test.js
d1ba7f5978277d0be328e6502191a2b024b13dd8c57ee65d161820fd01d18a26 ./test/ERC777Pool.test.js
01d0deee634f684b84d05a84fb39b2a2dc8a7caa0a09ffcb92c656f42ac68b40 ./test/MCDAwarePool.test.js
1de5f143e7fc499b42bc90286fc3865fe11cf3072bcf6e8924966e49f45c953f ./test/Pool.test.js
62b89bac62ef872ccaaae1eadc418dfc2f125123f942c918b7c7d8d61e04752d ./test/RecipientWhitelistERC777Pool.test.js
a4e1171bd92115cbcc927080d57a98dbb713d06f01765a6b492395b5bc216ff1 ./test/Stress.test.js
fd2e0ff47f71060a87195c78d41d2719863ce32e6651a104423ee836b3d967ef ./test/UniformRandomNumber.test.js
f881846c4579dcae4e3bde7d56755a10f73c39f916d5d50b86dd8742e7125f61 ./test/helpers/chai.js
13dc16b0b736a97e34c82c72066b2dd89a81a2653a187cf6e6c572cce8e299a4 ./test/helpers/constants.js
98513665566d8a633e88016f4ed49007e0ef48c0ed571c7cae0ca1d0867c67fe ./test/helpers/fromWei.js
2951899bffee2b26372958c236ad8dd12f86203342a70b5416d7be02f8614205 ./test/helpers/mineBlocks.js
056398003309ffa9fdd7f107464270647c2959118bfed0f498f63cf45de4ae9e ./test/helpers/PoolContext.js
7cb438ae93aea02bd2ba65b8d408e63159826d93300418f028917630172e69cb ./test/helpers/setupERC1820.js
4b313397adc10016a2f0843c5f2a05f466761f46ad79c7a13e2b3e5ccb10b4a2 ./test/helpers/toWei.js
```

About Quantstamp

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

Quantstamp's team boasts decades of combined experience in formal verification, static analysis, and software verification. Collectively, our individuals have over 500 Google scholar citations and numerous published papers. In its mission to proliferate development and adoption of blockchain applications, Quantstamp is also developing a new protocol for smart contract verification to help smart contract developers and projects worldwide to perform cost-effective smart contract security audits.

To date, Quantstamp has helped to secure hundreds of millions of dollars of transaction value in smart contracts and has assisted dozens of blockchain projects globally with its white glove security auditing 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.

Finally, Quantstamp's dedication to research and development in the form of collaborations with leading academic institutions such as National University of Singapore and MIT (Massachusetts Institute of Technology) reflects Quantstamp's commitment to enable world-class smart contract innovation.

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. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The Solidity language itself and other smart contract languages remain under development and are subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity or the smart contract programming language, or other programming aspects that could present security risks. You may risk loss of tokens, Ether, and/or other loss. A report is not an endorsement (or other opinion) of any particular project or team, and the report does not guarantee the security of any particular project. A report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. To the fullest extent permitted by law, we disclaim all warranties, express 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 website, or any website or mobile application featured in any banner or other 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. You may risk loss of QSP tokens or other loss. 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.