October 19th 2019 — Quantstamp Verified

Pool Together 2.0

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

Executive Summary

Type	Protocol contracts
Auditors	Kacper Bąk, Senior Research Engineer Ed Zulkoski, Senior Security Engineer Jim Miller, Senior Blockchain Engineer
Timeline	2019-08-15 through 2019-09-24
EVM	Byzantium
Languages	Javascript, Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer Aided Verification, Manual Review
Specification	PoolTogether v0.2 Audit Checklist
Source Code	Repository Commit github.com/pt-contracts 5f3f3cb

Total Issues	6 (0 Resolved)	
High Risk Issues	0 (0 Resolved)	
Medium Risk Issues	0	
Low Risk Issues	1 (0 Resolved)	1 Issue
Informational Risk Issues	4	
Undetermined Risk Issues	0	

Overall Assessment

Overall, the code is well written and documented, follows best practices and makes a good use of managed dependencies. We have not found any significant issues. We provide ideas for further code and documentation improvements.

It is important to note that the the code assumes that the used Compound token contract is a "well-behaved" ERC20 token. Furthermore, although "rounding down" is the intended behavior, it is not clear that this is the best way to handle financial inputs.

The low severity issue, which is related to the use of `transfer()` instead of `call()` / `transfer()` may complicate gas strategy in future Ethereum upgrades.

Update: PoolTogether addressed our feedback as of commit `5f3f3f0`.

Severity Categories	
High	The issue puts a large number of users' sensitive information at risk, or is likely to lead to severe smart contract vulnerabilities that may cause significant financial implications for client and users.
Medium	The issue puts a subset of users' sensitive information at risk, would seriously disrupt the smart contract's intended execution, 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 vulnerability that the client should be aware of, but poses no immediate risk, but is relevant to security best practices or Defense in Depth.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or Defense in Depth.
Undetermined	The impact of the issue is uncertain.

Goals

- Can users funds get locked up in the contracts?
- Are calculations implemented correctly?
- Is the winner chosen fairly?

Changelog

- 2019-08-26 - Initial Report
- 2019-09-24 - Report updated based on commit `5f3f3f0`

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 overnights
- 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 specification, 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 executed 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, clarity, maintainability, security, and control based on the established industry and community practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Tutorial

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

Setup

Tool Setup:

- Truffle v5.0.28
- Ganache v6.5.0
- Mythril v0.20.1
- Security.email also 10x39x
- Node v8.16.2

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. Run the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
5. Flattened the source code using `truffle-flattener` to accommodate the auditing tools.
6. Installed the Mythril tool from Pipy: `pipenv install mythril`
7. Ran the Mythril tool on each contract: `myth -x path/to/contract`
8. Ran the Security.email `java -Dmx8096m -jar security-0.1.jar -ts contract.txt`
9. Installed Nodu 8 from here

Assessment

Findings

Centralization of Power

Security Low

Contract(s) affected: Pool.sol

Description: Smart contracts will often have an owner or admins to designate entities with special privileges to make modifications to the contract. Generally, the past admins have the privilege to control and open draws, set fee fractions, set beneficiary of the fees, and pause the contract in case of an emergency. Consequently, if the last admin is never removed, no new admin can ever be added.

Recommendation: Developers may take advantage of calling `renounceRoleAdmin()` in the following scenarios: 1) Admins can try to wait until a favorable block to call `renounceRoleAdmin()` since they control the entropy. 2) Admins can decide to never call `renounceRoleAdmin()`, in which case the interest is locked forever. 3) Admins can delay calling `renounceRoleAdmin()` for an arbitrary period of time, waiting for most other users to exit so that the odds of winning the lottery are heavily in their favor.

Recommendation: This centralization of power needs to be made clear to the users. If a new admin is being added to replace an old one, ensure that the new admin is added first and the old admin is removed after a reasonable delay or short before removing the old one. It is important to note, however, that the users can always receive their funds back, therefore the lottery is always "hooked" from their point of view.

Test Results

Test Suite Results

All the tests executed correctly.

```
Contract: DrawManager
  openNextDraw()
    when a previous draw has been committed and rewarded
      should open the next draw (71ms)
    when a previous draw has been committed but not rewarded
      should revert if it has errors
  commit()
    when ready to commit (2 blocks or less passed since open)
      when committed to the current open draw
        should emit the committed event (71ms)
        should set the committed draw
        should return false if the draw errors
  reward()
    when ready to reward (committed to a previous open draw)
      should emit the rewarded event
      should set the last rewarded draw to the committed draw (71ms)
      should mint the winnings to the winner (71ms)
      should mint the fee to the beneficiary (71ms)
      should mint the previous balance eligibility and start a new open balance (101ms)
      should return false if there was no winner (71ms)
      should initialize the detection two blocks and update the open draw (120ms)
  canStartDraw()
    should return false when no draws exists
    after an open draw has been committed
      should return false if the open draw has passed the one open draw has passed
      should return true if there is no open draw only
  canCommitDraw()
    should return false when no draws exists
    when a committed draw exists:
      should return false if the block hash matches of the user
      and the user has deposited multiple times
      should return false if the block hash matches
  hasOwed()
    should return 0 if nothing exists
  withdraw()
    should allow the user to withdraw their open balance (49ms) gas)
    should emit the withdrawn event (71ms)
    should allow the user to withdraw all of their balances (121ms) gas)
  canRewardDraw()
    should return false(0) if no eligible deposits
  getWinningRandom()
    should return 0 if nothing exists
  getNextDraw()
    should allow the user to withdraw their open balance (49ms) gas)
    should emit the withdrawn event (71ms)
    should allow the user to withdraw all of their balances (121ms) gas)
  canRewardDraw()
    should return false(0) if no eligible deposits
  getWinningRandom()
    should return 0 if nothing exists
  getNextDraw()
    should return the next draw
  getOrderedSum()
    should return 0 if nothing exists
  getBalanceAt()
    should return the zero balance for the current draw (70ms)
    should return the zero balance for the current draw (70ms) gas)
  calculateCumulative()
    should revert if the cumulative balance is zero (70ms)
    should set an appropriate limit based on max integers (270ms)
  getOrderedSum()
    should return zero values if it does exists (270ms)
    should return the correct values for the current draw (270ms)
  with a fresh pool
    getOrderedSum()
      should fail if not enough values requested (249ms) gas)
      should return the correct sum values (270ms) gas)
      should allow multiple deposits (340ms gas)
  getOrderedCumulative()
      should contribute to the winnings (261ms gas)
  withdraw()
      should work for one participant (121ms gas)
      should work for two participants (121ms gas)
      should work after one such withdraw before the next draw (141ms) gas)
  with sponsorship
      should allow the sponsor to withdraw partially (49ms gas)
  balanceOf()
      should return the amount total to withdraw (101ms gas)
  when fee fraction is greater than zero
    should set the fee fraction (71ms gas)
  when a pool is rewarded without a winner
    should mint the fee to the beneficiary (71ms) (294ms)
  withdrawFeeTokens()
    should fail if the owner is not the next fee fraction (101ms gas)
    should not do anyone else is not the fee fraction (101ms gas)
  withdrawFeeTokens()
    should fail if the owner is not the next fee fraction (101ms gas)
    should not do if no more fee to set for the fee fraction (101ms gas)
  setFeeFraction()
    should not allow any more deposits (120ms) gas)
    should set the fee fraction (71ms gas)
  setBeneficiary()
    should allow deposits after unpausing (141ms) gas)

Contract: IDrawManager
  should work

Contract: IsContractWithFeeManager
  setFee()
    should revert if the sender tound is zero
    should skip the fixed 4 numbers that cause module bloat
```

Code Coverage

The code features very good coverage. We recommend, however, improving branch coverage for `Pool.sol`.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	75.56	100	100	
DrawManager.sol	100	75.56	100	100	
Pool.sol	100	62.5	100	100	
UniCumulativeWeight.sol	100	100	100	100	
contracts/compound/	100	100	100	100	
ICEther.sol	100	100	100	100	
contracts/test/	92.86	50	90.91	92.86	
CToken.sol	81.71	50	75	81.71	45,65
ExposedDrawManager.sol	100	100	100	100	
ExposedUniCumulativeWeight.sol	100	100	100	100	
Token.sol	100	100	100	100	
All files	**99.06**	**73.47**	**97.22**	**99.08**	

Automated Analyses

Mythril

Mythril reported External Call To Fixed Address and External Call To User-Supplied Address in the following cases in `Pool.sol`:

```
- The function `_commit()` calls cToken.balanceWithInvest().

- The function `_deposit()` calls token().transfer(msg.sender), token().approve(), and cToken.mint().  If the Compound ERC20 token is trusted, these warnings may be regarded as false positives.
```

Security

Security did not report any issues.

Adherence to Specification

The code mostly adheres to the presented specification. Minor deviations:

- **Update:** Fixed in `Pool.sol`. description of the function `depositSponsorship()` says that "There must be an open draw to deposit to". It does not appear to be enforced. We recommend adding `requireOpenDraw` modifiers to the function to implement this requirement.
- **Update:** Fixed in `Pool.sol`. Line 375, with check if `amount > 0`, however the error message suggests otherwise. We recommend updating the error message to say that the amount is not greater than zero.
- **Update:** Fixed in `Pool.sol`, line 512, the message should, perhaps, say "the beneficiary should not be 0x0".
- **Update:** Fixed in `Pool.sol`, in the function `reward()`, it seems like the line should contribute to the `feeBeneficiary` `sponsorshipBalance`, not `balance`, since the amount is not added to the staking pool via `drawState.deposit()`.

Code Documentation

The code is very well documented.

Adherence to Best Practices

The code mostly adhere to best practices. Minor improvements:

- **Update:** Fixed. Unused imports in `ExposedDrawManager.sol` and `ExposedUniCumulativeWeight.sol` to a test directory.
- **Update:** Fixed in `DrawManager.sol`, lines 73 and 80, instead of using magic numbers, we recommend using a named constant `MIN_FIELD_NODES = 4`. Furthermore, we recommend adding a comment why the number is was chosen.
- **Update:** Fixed in `DrawManager.sol`, line 204, `drawId()` is private but this 257 can be to used.
- **Update:** Fixed in `Pool.sol`, line 145, it is unnecessary to use `uint(1)` because of the check in the preceding line.
- **Update:** Fixed in `Pool.sol`, line 556, instead of using magic constant `0000000000000000000000000000000000000000`, we recommend using `address(0)`. Explicit type conversion is missing.
- **Update:** Fixed in `Pool.sol`, in `withdrawFeeTo()`, if fees VOD and VOX are reversed, you could reuse the the `balance` variable.

Appendix

File Signatures

The following are the SHA-256 hashes of the audited contracts and/or test files. A smart contract or file with a different SHA-256 hash has been modified, intentionally or otherwise, after the audit. 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 audit.

Contracts

```
6c68ff3f8f8d1d91ba1f5c7a3b8b3f5f32c7e0c97d03e6c9f8bc1f99e4d8ec
./contracts/DrawManager.sol
f7d8b8c3a7d9ba9a3a3f4f6d7ad4d8d1dc6a2ba7ec7f1b8b5f2c72f2f2
./contracts/Pool.sol
d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8
./contracts/test/ExposedDrawManager.sol
d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7
./contracts/test/ExposedUniCumulativeWeight.sol
a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0
./contracts/test/Token.sol
```

Test Suites

```
b4d8f7d8f8d2d8f8a2f8c5a3d8d3f5f32c7e0c97d03e6c9f8bc1f99e4d8ec
./test/DrawManager.test.js
f7d8b8c3a7d9ba9a3a3f4f6d7ad4d8d1dc6a2ba7ec7f1b8b5f2c72f2f2f2
./test/Pool.test.js
d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8d8
./test/UniCumulativeWeight.test.js
d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7d7
./test/helpers/IsContractWithFeeManager.test.js
a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0
./test/helpers/ICompound.test.js
```

About Quantstamp

Quantstamp is a 1st-generation leveled computing chain 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 50 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 paper, testing, and auditing services. As an evangelist of the blockchain ecosystem, Quantstamp assists core blockchain projects and leading companies, institutions 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 Blockchain Initiative of Technology aims 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 site 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 administrator of that site.

Quantstamp makes no representations or warranties as to the security or integrity of any connection or transmission made using the Internet or any other means, or which you for the purpose of Quantstamp causes to responsibility for the use of third party software on the website and shall not be liable to intentional viruses, any person or entity for any loss or damage in contract or tort, 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, whether things being examined may infringe upon intellectual property rights, nor does a report address the business or economic merits of any project or part thereof in any respect, 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 web site, or the internet. Without limiting the foregoing, we do not warrant that the content will meet the requirements of the client, others in the industry, or anyone else, or that the report will be free of any errors or other harmful components. Nothing contained in the report is intended to be construed as a representation or warranty by, or a basis for, or a recommendation to, any person or entity to enter into a contract, agreement, or business relationship, or otherwise to rely upon any product, service, or other asset. The foregoing limitations will be given effect to the extent permitted by law.

In no event, entities, guarantors, or assume responsibility for any consent or entities shall be liable for any direct or indirect or any other loss or damage, including without limitation, any loss of profit, business, contracts, revenue or goodwill, loss of anticipated savings or benefits, or any other monetary or pecuniary loss or damage. Moreover, the report should not be construed as a representation or warranty by, or a basis for, or a recommendation to, any person or entity 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 web site, or the internet. Without limiting the foregoing, we do not warrant that the content will meet the requirements of the client, others in the industry, or anyone else, or that the report will be free of any errors or other harmful components. Nothing contained in the report is intended to be construed as a representation or warranty by, or a basis for, or a recommendation to, any person or entity to enter into a contract, agreement, or business relationship, or otherwise to rely upon any product, service, or other asset. The foregoing limitations will be given effect to the extent permitted by law.

In no event, entities, guarantors or shall be liable for any consent or entities shall be liable for any direct or indirect or any other loss or damage, including without limitation, any loss of profit, business, contracts, revenue or goodwill, loss of anticipated savings or benefits, or any other monetary or pecuniary loss or damage.

Disclaimer

The 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 services, products, specifications, and documentation, will be at your sole risk and Quantstamp makes no representations or warranties of any kind, express or implied, including without limitation any implied warranties of merchantability, suitability, fitness for a particular purpose, or non-infringement, or that any of the foregoing will be uninterrupted or error-free. Cryptographic tokens are innovative and high risk in nature. You agree that Quantstamp shall have no responsibility for 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 not rely on a report in lieu of your own independent review of the subject matter. Quantstamp does not provide investment advice, or any other financial, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.