Quantstamp Security Assessment Certificate

August 5th 2021 — Quantstamp Verified

KeeperDAO Liquidity Pool and HidingVault

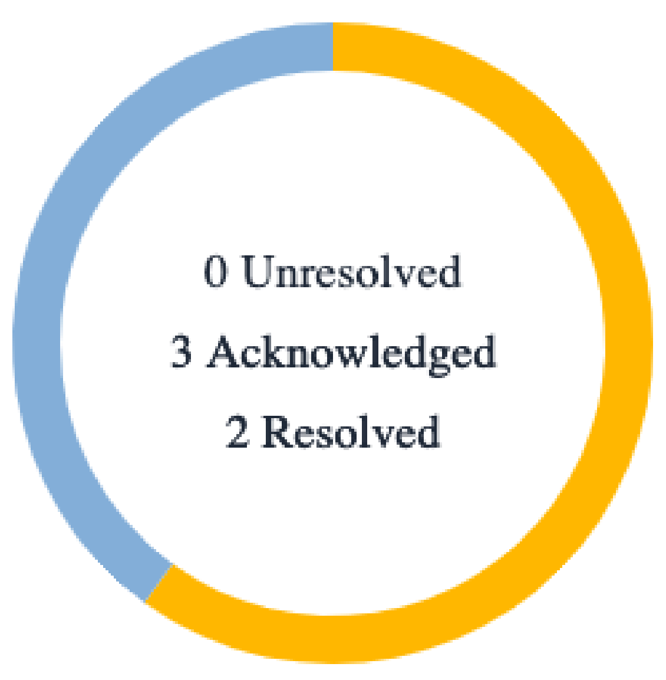
This audit report was prepared by Quantstamp, the leading blockchain security company.

Executive Summary

Type	DeFi protocol
Auditors	Kacper Bąk, Senior Research Engineer Poming Lee, Research Engineer Fayçal Lalidji, Security Auditor
Timeline	2021-07-26 through 2021-08-04
EVM	Berlin
Languages	Solidity, Typescript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Wiki
Documentation Quality	High
Test Quality	Medium
Source Code	Repository: protocol — Commit: 34b81da
Goals	• Can funds get locked up in the contract? • Can fund migration fail?

Repository	Commit
protocol	34b81da

Total Issues	**5** (2 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**5** (2 Resolved)
Informational Risk Issues	0 (0 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.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
? Undetermined	The impact of the issue is uncertain.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

We have reviewed the codebase and found a few issues of low severity. We recommend addressing all the issues and polishing the test suite. Some tests failed due to timeouts.
Update: the team has addressed all the issues as of commit `f41a98d`.

ID	Description	Severity	Status
QSP-1	Privileged Roles and Ownership	Low	Acknowledged
QSP-2	Interaction with External Contracts	Low	Acknowledged
QSP-3	Missing argument validation	Low	Acknowledged
QSP-4	`mintHidingVault()` is a `payable` function	Low	Fixed
QSP-5	Outstanding lent tokens are not considered by `migrate()`	Low	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.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

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

Setup

Tool Setup:

- Slither v0.8.0

Steps taken to run the tools:

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

Findings

QSP-1 Privileged Roles and Ownership

Severity: Low Risk

Status: Acknowledged

File(s) affected: `LiquidityPool.sol`, `HidingVaultNFT.sol`, `JITUCore.sol`, `JITUCompound.sol`, `KCompound.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. Specifically:

1. An operator of `LiquidityPool` can call the function `recoverTokens()` at any point in time, to retrieve from the contract any token that is not on the `recoverableTokensBlacklist`.
2. An operator of `LiquidityPool` can call the function `pause()` at any point in time, to pause all the withdrawal requests.
3. An owner of `HidingVaultNFT` can modify any function's code implementation of all outstanding vaults at will.

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

Update: The team informed us that they are in the process of setting up a DAO and all these roles would be transferred to that shortly.

QSP-2 Interaction with External Contracts

Severity: Low Risk

Status: Acknowledged

Description: The protocol relies on functionalities of external contracts. Therefore, security of the project depends on these contracts. While we are unaware of any immediate issues, it is important to note that defi protocols may be vulnerable to flash loan attacks, market manipulation, computational errors, etc. Furthermore, we also want to note that the project assumes constant supply of the external tokens, i.e., inflationary and deflationary tokens are not compatible.

Recommendation: We recommend reviewing external contracts to make sure they work as expected.

Update: The team acknowledged the finding. The team will make sure to do a lot of vetting before adding new assets and they tried to build their protocol in such a way that it has isolated risk for people who are using a specific protocol. E.g., someone not using Compound through their system would not lose any funds for any risk associated with Compound. Since the risk is isolated, and the people who are using that specific protocol are exposed to that specific protocol's risk they are not adding any risk to what users already take by using it.

QSP-3 Missing argument validation

Severity: Low Risk

Status: Acknowledged

File(s) affected: `JITUCore.sol`

Description: The constructor of `JITUCore` does not check if arguments are non-zero.

Recommendation: Add relevant checks.

QSP-4 `mintHidingVault()` is a `payable` function

Severity: Low Risk

Status: Fixed

File(s) affected: `HidingVaultNFT.sol`

Description: The function `mintHidingVault()` is marked as `payable`. Users may send ether erroneously.

Recommendation: Remove the `payable` attribute.

QSP-5 Outstanding lent tokens are not considered by `migrate()`

Severity: Low Risk

Status: Fixed

File(s) affected: `LiquidityPool.sol`

Description: The function `migrate()` does not take into account the outstanding tokens that were lent to adapters. While not an error in itself, this may lead to confusion or accounting issues if unnoticed by the operator. The new liquidity pool may have smaller balance than the original liquidity pool had initially. Furthermore, adapters may only repay funds to the original liquidity pool.

Recommendation: Unless this is intended, you can add checks that revert migration if there are still outstanding loans. Another possibility is to perform migration only if all the loans are repaid.

Automated Analyses

Slither

Slither reported the following:

- eth transfer to arbitrary user in `JITUCompound.sol#88`, `JITUCompound.sol#139`, and `LiquidityPool.sol#332`;
- uninitialized local variables in: `HidingVaultNFT.sol#62`, `LibCompound.sol#276`, and `LibCompound.sol#338`;
- `HidingVaultNFT.sol` being an upgradeable contract that does not protect its `initialize()` function.

We classified all issues besides the last one as false positives. **Update:** the team fixed the last issue as of commit `f41a98d`.

Adherence to Specification

1. `README.md`: "where `balanceOf` and `totalSupply` do not include the amount to be withdrawn, or the amount to be burned." => According to the code implementation, they do include the amount to be withdrawn, and the amount to be burned.
2. `LiquidityPool.sol#237`: the comment should be "inter-blocks" instead of "intra block", because based on the implementation repays may happen after multiple blocks.
3. For `LiquidityPool.sol`, please confirm if renouncing the `msg.sender`'s role as an operator for `_newLP` at the end of the function `migrate` is intended.
4. `LibCompound.migrate()`: please confirm if `L59`-`L62` is intended.
5. `LibCompound.mulExp()` does not look like an implementation of an exponential function calculation. Please confirm if this is intended.

Adherence to Best Practices

1. In `JITUCompound.protect()` there is no check if `unprotected[_vault]` is already `false`.
2. `LibCompound.preempt()`: consider adding a `require` statement to check and revert if `seizeTokens` obtained in `L112` is zero.
3. `LibCompound.collateralValueInUSD`: consider reverting the transaction if `exchangeRate` or `collateralFactor` is zero.

Test Results

Test Suite Results

All tests but one passed. The one test failed due to timeout.

```
HidingVault
   ✓ Should set the correct owner
   ✓ Should set the correct ERC721 name and symbol
   ✓ Should be able to mint a compound position
   ✓ Should be able to recover tokens
   ✓ Should fail to mint two compound positions with the same salt

KCompound
   ✓ Should be able to mint deterministic positions
   ✓ Should be able to mint a compound position
   ✓ Should return the correct address as beneficiary
   ✓ Should fail to deposit not registered token
   ✓ Should be able to deposit ETH (100000000000000)
   ✓ Should be able to deposit DAI (1000000000000000000000)
   ✓ Should fail to borrow not registered token
   ✓ Should be able to borrow DAI (10000000000000)
   ✓ Should be able to borrow ETH (100000000000000)
   ✓ Should fail to repay not registered token
   ✓ Should be able to repay DAI
   ✓ Should be able to repay ETH
   ✓ Should fail to withdraw not registered token
   ✓ Should be able to withdraw ETH (50000000000000)
   ✓ Should be able to withdraw DAI (500000000000000000000)
   ✓ Should be able to exit and enter a market
   ✓ Should be able to migrate a position
   ✓ Should be able to migrate a position
   ✓ ALICE can transfer her position to BOB

KCompound - gelatojitu
   ✓ Should be able to deposit ETH
   ✓ Should be able to withdraw eth from contract
   ✓ Should be able to withdraw tokens from contract
   ✓ Should fail to withdraw if not owner
   ✓ Should have ETH has payment token
   ✓ Should be able to set payment token
   ✓ Should fail to set payment tokens if not owner
   ✓ Should be able to preempt by providing buffer (Collateral: ETH)
   ✓ Should be able to preempt by providing buffer (Collateral: DAI)

   ✓ Keeper can't underwrite user's compound position before underwriter provide assets
   ✓ Keeper can underwrite user's compound position after underwriter provide assets
   ✓ Keeper can reclaim buffer if the user's compound position is above water
   ✓ Keeper can't preempt the liquidation if the user's compound position is not liquidation
   ✓ Keeper can preempt the liquidation only if the user's compound position is below the liquidation threshold
   ✓ Keeper can reclaim buffer once the user's compound position is liquidated
   ✓ Keeper can preempt the liquidation : ETH

User Story
   ✓ User can initialize compound position
   ✓ BOB can migrate a position that does not have ETH in it
   ✓ User can migrate his compound position
   ✓ User can deposit ETH or any supported ERC20
   ✓ User can borrow ETH or any supported ERC20 (user can specify the recipient who will get the borrowed tokens)
   ✓ User can repay borrowed tokens
   ✓ User can withdraw ETH or any supported ERC20 (user can specify the recipient who will get the withdrawl tokens)

LiquidityPoolV1
   ✓ Can get SimpleLP's Operator
   ✓ Should get the correct kToken address
   ✓ Should not register the same token twice
   ✓ Should not deposit with an non registered token
   ✓ Should receive correct amount of kTokens on first deposit to SimpleLP
   ✓ Should receive correct amount of kTokens on first deposit to SimpleLP
   ✓ Should receive correct amount of kTokens on second deposit to SimpleLP
   ✓ Should not be able to send ETH during an ERC20 deposit
   ✓ Should receive correct amount of kTokens on second deposit to SimpleLP
   ✓ Should fail to deposit when the eth amount is inconsistent
   ✓ Should not allow deposits when paused
   ✓ Should not flash lend with an non registered token
   ✓ Can flash lend wether
   ✓ Can flash lend ether
   ✓ Should not lend ether when paused
   ✓ Should revert when the borrowerfn reverts
   ✓ Can revert on an invalid flash loan
   ✓ Should be able to migrate
   ✓ Should be able to migrate even when the LP is paused
   ✓ Can withdraw partial wEther from SimpleLP
   ✓ Can withdraw partial ether from SimpleLP
   ✓ Should not allow withdrawals when paused
   ✓ Can withdraw all wEther from SimpleLP
   ✓ Can withdraw all Ether from SimpleLP
   ✓ Should not withdraw Ether from SimpleLP when there are no kTokens
   ✓ Should successfully handle profit re-distribution when the profit can be evenly split
   ✓ Should handle profit re-distribution when the balance profit cannot be evenly split
   ✓ Non operator should not be able to migrate
   ✓ Should not withdraw with an unregistered token
   ✓ Should not be able to recover liquidity tokens

LiquidityPoolV2
   ✓ Operator should be able to update deposit fee
   ✓ A non-operator should not be able to update deposit fee
   ✓ Fee cannot exceed 10000
   ✓ Can deposit Ether with fees
   ✓ Can deposit ERC20s with fees

LiquidityPoolV4: Borrow With Pool Fees
   ✓ Operator should be able to update deposit fee
   ✓ A non-operator should not be able to update deposit fee
   ✓ Fee cannot exceed 10000
   ✓ Operator should be able to update fee pool
   ✓ A non-operator should not be able to update deposit fee
   ✓ Should send the correct amount of eth to the fee pool when pool fee is 100%
```

```
    ✓ Should send the correct amount of eth to the fee pool when pool fee is 40%
    ✓ Should send the correct amount of eth to the fee pool when pool fee is 0%
    ✓ Should send the correct amount of ERC20 tokens to the fee pool when pool fee is 100%
    ✓ Should send the correct amount of ERC20 tokens to the fee pool when pool fee is 40%
    ✓ Should send the correct amount of ERC20 tokens to the fee pool when pool fee is 60%
    ✓ Should send the correct amount of ERC20 tokens to the fee pool when pool fee is 0%
    ✓ Should revert if a non-LP address calls the borrowerProxy

  LiquidityPoolV4
    ✓ Operator should be able to update deposit fee
    ✓ A non-operator should not be able to update deposit fee
    ✓ Fee cannot exceed 10000
    ✓ Can deposit Ether with fees
    ✓ Can deposit ERC20s with fees
    ✓ Adapter can borrow funds and repay in installments
    ✓ Adapter can borrow funds and repay with profit

  102 passing (8m)
  1 failing

  1) KCompound - jitu
       "before all" hook for "Should be able to deposit eth":
     Error: Timeout of 60000ms exceeded. For async tests and hooks, ensure "done()" is called; if returning a Promise, ensure it resolves. (/Users/kbak/Downloads/protocol-
f41a98dab3111e704bda7e51fc1c5eaba647adf0/test/integrations/compound/unit-tests/underwriter.ts)
      at listOnTimeout (internal/timers.js:531:17)
      at processTimers (internal/timers.js:475:7)
```

Code Coverage

Overall, the coverage is good, however, we recommend improving the test suite to achieve at least 80% coverage in each category. It is important to note that one test fails during a regular test suite execution but in passes in the coverage mode.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
hidingvault/	92.86	57.14	84.62	93.33	
HidingVault.sol	90	50	100	90.91	41
HidingVaultNFT.sol	94.12	100	75	94.12	78
LibHidingVault.sol	100	100	100	100	
Proxy.sol	100	100	100	100	
hidingvault/plugins/	100	60	100	100	
JITUCore.sol	100	60	100	100	
hidingvault/plugins/compound/	85.78	60.16	70.15	84.06	
Compound.sol	75	100	60	75	201,202,207
IJITUCompound.sol	100	100	100	100	
IKCompound.sol	100	100	100	100	
JITUCompound.sol	100	61.54	100	100	
KCompound.sol	0	0	0	0	… 169,176,183
LibCToken.sol	100	75	100	100	
LibCompound.sol	100	65.22	100	100	
hidingvault/plugins/compound/gelato/	72.97	50	83.33	77.78	
GelatoJITU.sol	62.5	33.33	75	68.18	… 90,92,93,96
Gelatofied.sol	92.31	58.33	100	92.86	22
mock/	100	75	100	100	
MockKCompound.sol	100	75	100	100	
protocol/	96.3	73.61	88.89	96.4	
BorrowerProxy.sol	100	100	100	100	
CanReclaimTokens.sol	33.33	16.67	100	33.33	24,25,26,28
Interf.sol	100	100	100	100	
KRoles.sol	100	100	100	100	
LiquidityPool.sol	100	78.57	100	100	
Tokens.sol	92.31	50	60	92.86	38
All files	**89.64**	**63.95**	**82.66**	**89.57**	

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

```
14c97b56f3bc2e347cbbb34bbbfa9e2415076d9a659af222f7801cd6edb199e6 ./LiquidityPool.sol
0a0a9a04d721461281f7351f9eea86a68724398877c59785bebc9cdd02b8531d ./hidingvault/HidingVaultNFT.sol
3e36579418936c12edd74aaa385ffbc31c182d52076603d02f002c87a1ad2b12 ./hidingvault/HidingVault.sol
b505655d01ca07c79693e9e0b8cac418448b84c023357009236d1123435727d0 ./hidingvault/Proxy.sol
0999d7fe4acc97174202ce6b6a784eea13187f291cecae20b3b8773a82cb3b32 ./hidingvault/LibHidingVault.sol
dd31a1e311be8e50819b261ddeee800f0c5c0c1385db882171875bf7ad09498b ./hidingvault/plugins/JITUCore.sol
38bcc5fd0c53ff9508807d8bde2baac674e3cb2b0793c243c0de29db3a28cc34 ./hidingvault/plugins/compound/JITUCompound.sol
123803ce8481eb8f5afd17de2b20bd6d6f1c7d59a2c29391298ef5c1a9ef3fc9 ./hidingvault/plugins/compound/IJITUCompound.sol
d72227972ff534c24d611d31d75187099eec653553851d439db9a3af02791591 ./hidingvault/plugins/compound/LibCompound.sol
5c8ae12628ad69965c2359359db843803e325b5421691df0a6ccf16662d71b48 ./hidingvault/plugins/compound/LibCToken.sol
bdfd276df561b6a62156f23700fa93ab891679c7aaf445184cc2201df0da7d21 ./hidingvault/plugins/compound/IKCompound.sol
0e326892ae41e92b2b982c2fea4b805481281d5555e9d8bf2ba25c6a1215c4e8 ./hidingvault/plugins/compound/KCompound.sol
```

Tests

```
4487d1b6db24fd8975c113d30149322a3e89b47fd35d821cca68313064b08250 ./test/helper.ts
d79b7439cc67a5c79a667801d72a889c761670a7d4eea9d3de251efe03b8bf6a ./test/hidingvault/hiding-vault.ts
98b02681e6ef3a85e047ea897660fe424767693bf380ef2ba6d5aab0e060ea01 ./test/integrations/compound/helper.ts
cad64d95cb28be3b16ac5f419cf9c6a9267f596482dff9fde0746fc4696b4279 ./test/integrations/compound/user-stories/keeper.ts
59946642fe7c599e15dc91ca608be71cb502781d481bc6ec34b4b514ef4ed2c3 ./test/integrations/compound/user-stories/user.ts
9f79ba929070c7da33b71760c39629f249c8cf503186d522cf3fe7be3903df8f ./test/integrations/compound/unit-tests/underwriter.ts
c84ed5a89f62d76d5c788e523da5dd21db5068c796b8147f4a2f92e2d374b20b ./test/integrations/compound/unit-tests/gelato-jitu.ts
9f7ae3cc6f24dad3dd2aef90bbabd49b77413ca328f80a083caa54304e999582 ./test/integrations/compound/unit-tests/compound-position.ts
cd73c21d2171df0706d2e57ad2d4211420064c974a4a61a2dd6c53c5eafe7304 ./test/protocol/liquidity-pool-v2.ts
be96cc45ef87fcde692e17b7fd70fd2d3813d978354a5e995d4558064c22b5f2 ./test/protocol/liquidity-pool-v4.ts
eb217553f1cc4c3a9458475302d4d872c8902ea086f6643c27d766eb0bc6030f ./test/protocol/liquidity-pool-v1.ts
```

Changelog

- 2021-08-03 - Initial report
- 2021-08-04 - Revised report based on commit `f41a98d`.

About Quantstamp

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

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.