Quantstamp Security Assessment Certificate

March 29th 2021 — Quantstamp Verified

Saddle Finance VirtualSwap

This security assessment was prepared by Quantstamp, the leader in blockchain security

Executive Summary

Type	Bridge and SynthSwap Contracts
Auditors	Ed Zulkoski, Senior Security Engineer Jose Ignacio Orlicki, Senior Engineer
Timeline	2021-02-22 through 2021-03-30
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Virtual Swap Design Doc SIP-89
Documentation Quality	High
Test Quality	Medium

Source Code

Repository	Commit
saddle-contract	5452b15 (initial report)
saddle-contract	b0c731c (final report)

Total Issues	**5** (5 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (1 Resolved)
Low Risk Issues	0 (0 Resolved)
Informational Risk Issues	**2** (2 Resolved)
Undetermined Risk Issues	**2** (2 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

This audit only pertains to the VirtualSwap `Bridge` and `SynthSwapper` contracts. The Saddle VirtualSwap contracts appear to be well-architected and properly documented. During our audit, several issues of ranging severity were uncovered, including a mismatch between the inline specification and code, as well as potential access-control issues. We recommend addressing all issues before deploying to mainnet.
Inline documentation and tests are in general of high quality. We recommend adding coverage scripts for the VirtualSwap contracts, however we acknowledge that the mainnet dependency for this test suite makes this non-trivial.
Update: all issues have been resolved as of commit b0c731ce.

ID	Description	Severity	Status
QSP-1	Mismatch between inline specification and `withdraw` function	^ Medium	Fixed
QSP-2	Unchecked function return values	○ Informational	Fixed
QSP-3	Settle cloned swappers can be destroyed	○ Informational	Fixed
QSP-4	`completeToSynth` is callable by any user	? Undetermined	Fixed
QSP-5	States `ReadyToSettle`, `PartiallyCompleted` and `Completed` are never checked and can lead to abnormal calls.	? Undetermined	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 v0.6.14
- Mythril v0.22.16

Steps taken to run the tools:

Installed the Slither tool: `pip install slither-analyzer` Run Slither from the project directory: `slither .` Installed the Mythril tool from Pypi: `pip3 install mythril` Ran the Mythril tool on each contract: `myth -x path/to/contract`

Findings

QSP-1 Mismatch between inline specification and `withdraw` function

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `Bridge.sol`

Description: The comment on L275-276 states: "Reverts if the given `itemId` does not represent a `synthToToken` or a `tokenToSynth` swap." However, the function checks `swapType > PendingSwapType.TokenToSynth`, which allows either `SynthToToken` or `TokenToToken` swap types.

Recommendation: Revise either the `require` statement or the inline comment.

QSP-2 Unchecked function return values

Severity: *Informational*

Status: Fixed

File(s) affected: `SynthSwapper.sol`

Description: In order to ensure that the functions updated state properly, it should be checked that `transfer` calls in `swapSynthToToken` and `withdraw` return true.

Recommendation: Since some tokens are not strictly ERC20-compliant and may not return a boolean, consider using OpenZeppelin's SafeERC20.

QSP-3 Settle cloned swappers can be destroyed

Severity: *Informational*

Status: Fixed

File(s) affected: `SynthSwapper.sol, synthetix/contracts/ExchangeState.sol`

Description: To save some gas on settlement and storage in general (in case of backup or migration), a cloned `SynthSwapper` that has already been used for settled swaps, can be destroyed. You can also remove the swaps from the pending list.

Recommendation: Use `selfdestruct()` to destruct the `SynthSwapper` contracts after the `_settle()` settlement, as the last thing to do, on functions 'withdraw()', `completeToSynth()` and `completeToToken()`. Optionally, you can also remove the completed swaps from the pending list `pendingSynthSwaps`.

QSP-4 `completeToSynth` is callable by any user

Severity: *Undetermined*

Status: Fixed

File(s) affected: `Bridge.sol`

Description: The function `completeToSynth` is callable by any user after the `maxSecsLeftInWaitingPeriod`. Although the `synth` wil be sent to the correct `nftOwner` regardless of `msg.sender`, this could have impact on external contracts that rely upon calling `completeToSynth`. For example, suppose some contract has a function of the form:

```
function getSynthAndTransfer(...) {
    Bridge.completeToSynth(id);
    synth.transfer(external_addr);
}
```

If another arbitrary user invokes `completeToSynth` before the contract invokes it, the funds could be locked forever (assuming the contract did not have sufficient "retrieval" functions).

Recommendation: Consider adding a `require(nftOwner == msg.sender, "not owner");` in `completeToSynth`.

QSP-5 States `ReadyToSettle`, `PartiallyCompleted` and `Completed` are never checked and can lead to abnormal calls.

Severity: *Undetermined*

Status: Fixed

File(s) affected: `Bridge.sol, SynthSwapper.sol`

Description: These pending swap states are set but never checked anywhere; this must be due to implicit assumptions on the state of swaps, but opens the door for duplicated calls to, for example, `completeToToken()`. A second duplicated call to `completeToToken()` might be possible with `swapAmount = 0` even if it was completed before.

Recommendation: Use checks for pending swaps states and also check that `swapAmount` is greater than zero. Similar for `synthInAmount` and `tokenInAmount`, and all functions that depend on swap states.

Automated Analyses

Slither

Slither warns that `_setPendingSwapState` performs multiplication on the result of division, however this is intended semantics for the function and no precision issues will occur.

Mythril

Myth warned of several calls to user-supplied token addresses in `SynthSwapper.sol`, however since this contract is restricted to only be called from the `Bridge`, these were classified as false positives.

Adherence to Specification

The code appears to adhere to provided specifications.

Code Documentation

1. [Fixed] It should be noted that `Bridge._setPendingSwapType` should not be called after `Bridge._setPendingSwapState` for an `itemId`, as it would erase the state. While this does not happen in the code currently, it may be worth adding a comment to these functions In order to mitigate future issues.
2. [Fixed] In `Bridge.sol` on L517, `TokenToSynth` should be `SynthToToken`.

Adherence to Best Practices

1. [Fixed] The function `Bridge._settle` invokes `maxSecsLeftInWaitingPeriod`. While this is fine as is, it appears that `exchanger._internalSettle` also invokes `maxSecsLeftInWaitingPeriod`, and therefore some gas could potentially be saved by removing the duplicate check. Note however that this check could be desirable if the `exchanger` is updated to not include the check.
2. [Fixed] Consider improving compressed variables and field names such as `pstss` and `ss`.

Test Results

Test Suite Results

```
  Allowlist
    setPoolCap, getPoolCap
deployed USD pool clone (targeting "SwapFlashLoan") at 0x856e4424f806D16E8CBC702B3c0F2ede5468eae5
┌─────────┬──────────────────────┬──────────────────────────────────────────────┐
│ (index) │          0           │                      1                       │
├─────────┼──────────────────────┼──────────────────────────────────────────────┤
│    0    │      'Allowlist'     │ '0x5FbDB2315678afecb367f032d93F642f64180aa3' │
│    1    │      'MathUtils'     │ '0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512' │
│    2    │  'SwapUtilsGuarded'  │ '0x9fE46736679d2D9a65F0992F2272dE9f3c7fa6e0' │
│    3    │      'SwapUtils'     │ '0xCf7Ed3AccA5a467e9e704C703E8D87F634fB0Fc9' │
│    4    │    'SwapDeployer'    │ '0xDc64a140Aa3E981100a9becA4E685f962f0cF6C9' │
│    5    │         'Swap'       │ '0x5FC8d32690cc91D4c39d9d3abcBD16989F875707' │
│    6    │   'SwapFlashLoan'    │ '0x0165878A594ca255338adfa4d48449f69242Eb8F' │
│    7    │         'TBTC'       │ '0xa513E6E4b8f2a923D98304ec87F64353C4D5C853' │
│    8    │         'WBTC'       │ '0x8A791620dd6260079BF849Dc5567aDC3F2FdC318' │
│    9    │        'RENBTC'      │ '0xB7f8BC63BbcaD18155201308C8f3540b07f84F5e' │
│   10    │         'SBTC'       │ '0x0DCd1Bf9A1b36cE34237eEaFef220932846BCD82' │
│   11    │   'SaddleBTCPool'    │ '0x0B306BF915C4d645ff596e518fAf3F9669b97016' │
│   12    │ 'SaddleBTCPoolLPToken' │ '0x524F04724632eED237cbA3c37272e018b3A7967e' │
│   13    │         'DAI'        │ '0x9A9f2CCfdE556A7E9Ff0848998Aa4a0CFD8863AE' │
│   14    │         'USDC'       │ '0x3Aa5ebB10DC797CAC828524e59A333d0A371443c' │
│   15    │         'USDT'       │ '0x59b670e9fA9D0A427751Af201D676719a970857b' │
│   16    │   'SaddleUSDPool'    │ '0x856e4424f806D16E8CBC702B3c0F2ede5468eae5' │
│   17    │ 'SaddleUSDPoolLPToken' │ '0x63f84713F52422Af2F8E18b56703b0f80CCcCBcE' │
└─────────┴──────────────────────┴──────────────────────────────────────────────┘
    ✓ Emits PoolCap event
    ✓ Reverts when the pool address is 0x0
    ✓ Reverts when non-owner tries to set the pool cap
    ✓ Sets and gets pool cap
  setPoolAccountLimit and getPoolAccountLimit
    ✓ Emits PoolAccountLimit event
    ✓ Reverts when the pool address is 0x0
    ✓ Reverts when non-owner tries to set the pool account limit
    ✓ Sets and gets pool account limit
  verifyAddress() & isAccountVerified()
    ✓ Returns true when proof and address are correct
    ✓ Returns true when merkleProof is empty but the account has been verified
    ✓ Returns true when merkleProof is wrong but the account has been verified
    ✓ Returns false when merkleProof is empty and the account has NOT been verified
    ✓ Returns false when address is wrong
  updateMerkleRoot
    ✓ Emits NewMerkleRoot event
    ✓ Updates merkleRoot successfully
    ✓ Reverts when called by non-owner

Swap Flashloan
  ✓ Reverts when the borrower does not have enough to pay back
  ✓ Reverts when flashloan debt is not paid
  ✓ Reverts when calling re-entering swap contract via `addLiquidity`
  ✓ Reverts when calling re-entering swap contract via `swap`
  ✓ Reverts when calling re-entering swap contract via `removeLiquidity`
  ✓ Reverts when calling re-entering swap contract via `removeLiquidityOneToken`
  ✓ Succeeds when fee is paid off
  setFlashLoanFees
    ✓ Reverts when called by non-owner
    ✓ Reverts when fees are not in the range
    ✓ Succeeds when fees are in the valid range

GenericERC20
  ✓ Reverts when minting 0

LPToken
  ✓ Reverts when minting 0

MathUtils
  within1
    ✓ Returns true when a > b and a - b <= 1
    ✓ Returns false when a > b and a - b > 1
    ✓ Returns true when a <= b and b - a <= 1
    ✓ Returns false when a <= b and b - a > 1
    ✓ Reverts during an integer overflow
  difference
    ✓ Returns correct difference when a > b
    ✓ Returns correct difference when a <= b
    ✓ Reverts during an integer overflow

StakeableTokenWrapper
  ✓ Reverts when staking 0
  ✓ Emits an event on staking
  ✓ Emits an event on withdrawing
  ✓ Only allows staked funds to be withdrawn
  ✓ Returns correct staked balances
  ✓ Returns correct total supply

Swap
  swapStorage
    lpToken
      ✓ Returns correct lpTokenName
      ✓ Returns correct lpTokenSymbol
      ✓ Returns true after successfully calling transferFrom
    A
      ✓ Returns correct A value
    fee
      ✓ Returns correct fee value
    adminFee
      ✓ Returns correct adminFee value
  getToken
    ✓ Returns correct addresses of pooled tokens
    ✓ Reverts when index is out of range
  getTokenIndex
    ✓ Returns correct token indexes
    ✓ Reverts when token address is not found
  getTokenBalance
    ✓ Returns correct balances of pooled tokens
    ✓ Reverts when index is out of range
  getA
    ✓ Returns correct value
  addLiquidity
    ✓ Reverts when contract is paused
    ✓ Reverts with 'Amounts must match pooled tokens'
    ✓ Reverts with 'Cannot withdraw more than available'
```

```
      ✓ Reverts with 'Must supply all tokens in pool'
      ✓ Succeeds with expected output amount of pool tokens
      ✓ Succeeds with actual pool token amount being within ±0.1% range of calculated pool token
      ✓ Succeeds with correctly updated tokenBalance after imbalanced deposit
addLiquidity: Expected 2997459774673651937, got 2997459774673651937
      ✓ Returns correct minted lpToken amount
      ✓ Reverts when minToMint is not reached due to front running
      ✓ Reverts when block is mined after deadline
      ✓ Emits addLiquidity event
    removeLiquidity
      ✓ Reverts with 'Cannot exceed total supply'
      ✓ Reverts with 'minAmounts must match poolTokens'
      ✓ Succeeds even when contract is paused
      ✓ Succeeds with expected return amounts of underlying tokens
addLiquidity: Expected 2997459774673651937, got 2997459774673651937
removeLiquidity: Expected 1199593357354995187, got 1199593357354995187
removeLiquidity: Expected 1799390036032492780, got 1799390036032492780
      ✓ Returns correct amounts of received tokens
      ✓ Reverts when user tries to burn more LP tokens than they own
      ✓ Reverts when minAmounts of underlying tokens are not reached due to front running
      ✓ Reverts when block is mined after deadline
      ✓ Emits removeLiquidity event
    removeLiquidityImbalance
      ✓ Reverts when contract is paused
      ✓ Reverts with 'Amounts should match pool tokens'
      ✓ Reverts with 'Cannot withdraw more than available'
      ✓ Succeeds with calculated max amount of pool token to be burned (±0.1%)
addLiquidity: Expected 2997459774673651937, got 2997459774673651937
removeLiquidityImbalance: Expected 1110205039594566418, got 1110205039594566418
      ✓ Returns correct amount of burned lpToken
      ✓ Reverts when user tries to burn more LP tokens than they own
      ✓ Reverts when minAmounts of underlying tokens are not reached due to front running
      ✓ Reverts when block is mined after deadline
      ✓ Emits RemoveLiquidityImbalance event
    removeLiquidityOneToken
      ✓ Reverts when contract is paused.
      ✓ Reverts with 'Token index out of range'
      ✓ Reverts with 'Withdraw exceeds available'
      ✓ Reverts with 'Token not found'
      ✓ Succeeds with calculated token amount as minAmount
addLiquidity: Expected 2997459774673651937, got 2997459774673651937
removeLiquidityOneToken: Expected 1864597634420375471, got 1864597634420375471
      ✓ Returns correct amount of received token
      ✓ Reverts when user tries to burn more LP tokens than they own
      ✓ Reverts when minAmount of underlying token is not reached due to front running
      ✓ Reverts when block is mined after deadline
      ✓ Emits RemoveLiquidityOne event
    swap
      ✓ Reverts when contract is paused
      ✓ Reverts with 'Token index out of range'
      ✓ Reverts with 'Cannot swap more than you own'
      ✓ Succeeds with expected swap amounts
      ✓ Reverts when minDy (minimum amount token to receive) is not reached due to front running
      ✓ Succeeds when using lower minDy even when transaction is front-ran
addLiquidity: Expected 2997459774673651937, got 2997459774673651937
swap: Expected 999000000000000000, got 999000000000000000
      ✓ Returns correct amount of received token
      ✓ Reverts when block is mined after deadline
      ✓ Emits TokenSwap event
    getVirtualPrice
      ✓ Returns expected value after initial deposit
      ✓ Returns expected values after swaps
      ✓ Returns expected values after imbalanced withdrawal
      ✓ Value is unchanged after balanced deposits
      ✓ Value is unchanged after balanced withdrawals
    setSwapFee
      ✓ Emits NewSwapFee event
      ✓ Reverts when called by non-owners
      ✓ Reverts when fee is higher than the limit
      ✓ Succeeds when fee is within the limit
    setAdminFee
      ✓ Emits NewAdminFee event
      ✓ Reverts when called by non-owners
      ✓ Reverts when adminFee is higher than the limit
      ✓ Succeeds when adminFee is within the limit
    getAdminBalance
      ✓ Reverts with 'Token index out of range'
      ✓ Is always 0 when adminFee is set to 0
      ✓ Returns expected amounts after swaps when adminFee is higher than 0
    withdrawAdminFees
      ✓ Reverts when called by non-owners
      ✓ Succeeds when there are no fees withdrawn
      ✓ Succeeds with expected amount of fees withdrawn
      ✓ Withdrawing admin fees has no impact on users' withdrawal
  Test withdrawal fees on removeLiquidity
    ✓ Removing liquidity immediately after deposit
    ✓ Removing liquidity 2 weeks after deposit
    ✓ Removing liquidity 4 weeks after deposit
  Test withdrawal fees on removeLiquidityOne
    ✓ Removing liquidity immediately after deposit
    ✓ Removing liquidity 2 weeks after deposit
    ✓ Removing liquidity 4 weeks after deposit
  Test withdrawal fees on removeLiquidityImbalance
    ✓ Removing liquidity immediately after deposit
    ✓ Removing liquidity 2 weeks after deposit
    ✓ Removing liquidity 4 weeks after deposit
  Verify changing withdraw fee works as expected
    ✓ Increase withdraw fee from 0% to 0.5%, immediately after last deposit
    ✓ Increase withdraw fee from 0% to 0.5%, 2 weeks after last deposit
    ✓ Increase withdraw fee from 0% to 0.5%, 4 weeks after last deposit
    ✓ Increase withdraw fee from 0.5% to 1%
    ✓ Decrease withdraw fee from 0.5% to 0%
    ✓ Decrease withdraw fee from 1% to 0.5%
  updateUserWithdrawFee
    ✓ Reverts with 'Only callable by pool token'
    ✓ Test adding liquidity, and once again at 2 weeks mark then removing all deposits at 4 weeks mark
    ✓ Verify withdraw fees are updated on transfer
  setDefaultWithdrawFee
    ✓ Emits NewWithdrawFee event
    ✓ Setting the withdraw fee affects past deposits as well
    ✓ Reverts when fee is too high
  rampA
    ✓ Emits RampA event
    ✓ Succeeds to ramp upwards
    ✓ Succeeds to ramp downwards
    ✓ Reverts when non-owner calls it
    ✓ Reverts with 'Wait 1 day before starting ramp'
    ✓ Reverts with 'Insufficient ramp time'
    ✓ Reverts with 'futureA_ must be > 0 and < MAX_A'
    ✓ Reverts with 'futureA_ is too small'
    ✓ Reverts with 'futureA_ is too large'
  stopRampA
    ✓ Emits StopRampA event
    ✓ Stop ramp succeeds
    ✓ Reverts with 'Ramp is already stopped'
  Check for timestamp manipulations
    ✓ Check for maximum differences in A and virtual price when A is increasing
    ✓ Check for maximum differences in A and virtual price when A is decreasing
    Check for attacks while A is ramping upwards
      When tokens are priced equally: attacker creates massive imbalance prior to A change, and resolves it after
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack fails with 2 weeks between transactions (mimics rapid A change)
      When token price is unequal: attacker 'resolves' the imbalance prior to A change, then recreates the imbalance.
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack succeeds with 2 weeks between transactions (mimics rapid A change)
    Check for attacks while A is ramping downwards
      When tokens are priced equally: attacker creates massive imbalance prior to A change, and resolves it after
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack succeeds with 2 weeks between transactions (mimics rapid A change)
      When token price is unequal: attacker 'resolves' the imbalance prior to A change, then recreates the imbalance.
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack fails with 2 weeks between transactions (mimics rapid A change)

Swap with 4 tokens
  addLiquidity
    ✓ Add liquidity succeeds with pool with 4 tokens
  swap
    ✓ Swap works between tokens with different decimals
  removeLiquidity
    ✓ Remove Liquidity succeeds
  Check for timestamp manipulations
    ✓ Check for maximum differences in A and virtual price when increasing
    ✓ Check for maximum differences in A and virtual price when decreasing
    Check for attacks while A is ramping upwards
      When tokens are priced equally: attacker creates massive imbalance prior to A change, and resolves it after
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack fails with 2 weeks between transactions (mimics rapid A change)
      When token price is unequal: attacker 'resolves' the imbalance prior to A change, then recreates the imbalance.
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack succeeds with 2 weeks between transactions (mimics rapid A change)
    Check for attacks while A is ramping downwards
      When tokens are priced equally: attacker creates massive imbalance prior to A change, and resolves it after
        ✓ Attack fails with 900 seconds between blocks
        ✓ Attack succeeds with 2 weeks between transactions (mimics rapid A change)
      When token price is unequal: attacker 'resolves' the imbalance prior to A change, then recreates the imbalance.
        ✓ Attack fails with 900 seconds between blocks
```

```
          ✓ Attack fails with 2 weeks between transactions (mimics rapid A change)

  Swap with 4 tokens
    addLiquidity
       ✓ Add liquidity succeeds with pool with 4 tokens
    swap
       ✓ Swap works between tokens with different decimals
    removeLiquidity
       ✓ Remove Liquidity succeeds
    Check for timestamp manipulations
       ✓ Check for maximum differences in A and virtual price when increasing
       ✓ Check for maximum differences in A and virtual price when decreasing
      Check for attacks while A is ramping upwards
        When tokens are priced equally: attacker creates massive imbalance prior to A change, and resolves it after
          ✓ Attack fails with 900 seconds between blocks
          ✓ Attack fails with 2 weeks between transactions (mimics rapid A change)
        When token price is unequal: attacker 'resolves' the imbalance prior to A change, then recreates the imbalance.
          ✓ Attack fails with 900 seconds between blocks
          ✓ Attack succeeds with 2 weeks between transactions (mimics rapid A change)
      Check for attacks while A is ramping downwards
        When tokens are priced equally: attacker creates massive imbalance prior to A change, and resolves it after
          ✓ Attack fails with 900 seconds between blocks
          ✓ Attack succeeds with 2 weeks between transactions (mimics rapid A change)
        When token price is unequal: attacker 'resolves' the imbalance prior to A change, then recreates the imbalance.
          ✓ Attack fails with 900 seconds between blocks
          ✓ Attack fails with 2 weeks between transactions (mimics rapid A change)

  Swap
    swapStorage#constructor
       ✓ Reverts with '_pooledTokens.length <= 1'
       ✓ Reverts with '_pooledTokens.length > 32'
       ✓ Reverts with '_pooledTokens decimals mismatch'
       ✓ Reverts with 'Duplicate tokens'
       ✓ Reverts with 'The 0 address isn't an ERC-20'
       ✓ Reverts with 'Token decimals exceeds max'
       ✓ Reverts with '_a exceeds maximum'
       ✓ Reverts with '_fee exceeds maximum'
       ✓ Reverts with '_adminFee exceeds maximum'
       ✓ Reverts with '_withdrawFee exceeds maximum'

  Virtual swap bridge [ @skip-on-coverage ]
    setSynthIndex
network block skew detected; skipping block events
deployed USD pool clone (targeting "SwapFlashLoan") at 0x8aCd85898458400f7Db866d53FCFF6f0D49741FF
```

(index)	0	1
0	'Allowlist'	'0x5FC8d32690cc91D4c39d9d3abcBD16989F875707'
1	'MathUtils'	'0x0165878A594ca255338adfa4d48449f69242Eb8F'
2	'SwapUtilsGuarded'	'0xa513E6E4b8f2a923D98304ec87F64353C4D5C853'
3	'SwapUtils'	'0x2279B7A0a67DB372996a5FaB50D91eAA73d2eBe6'
4	'SwapDeployer'	'0x8A791620dd6260079BF849Dc5567aDC3F2FdC318'
5	'Swap'	'0x610178dA211FEF7D417bC0e6FeD39F05609AD788'
6	'SwapFlashLoan'	'0xB7f8BC63BbcaD18155201308C8f3540b07f84F5e'
7	'TBTC'	'0xA51c1fc2f0D1a1b8494Ed1FE312d7C3a78Ed91C0'
8	'WBTC'	'0x9A676e781A523b5d0C0e43731313A708CB607508'
9	'RENBTC'	'0x959922bE3CAee4b8Cd9a407cc3ac1C251C2007B1'
10	'SBTC'	'0x68B1D87F95878fE05B998F19b66F4baba5De1aed'
11	'SaddleBTCPool'	'0xc6e7DF5E7b4f2A278906862b61205850344D4e7d'
12	'SaddleBTCPoolLPToken'	'0x553BED26A78b94862e53945941e4ad6E4F2497da'
13	'DAI'	'0x4ed7c70F96B99c776995fB64377f0d4aB3B0e1C1'
14	'USDC'	'0xa85233C63b9Ee964Add6F2cffe00Fd84eb32338f'
15	'USDT'	'0x7a2088a1bFc9d81c55368AE168C2C02570cB814F'
16	'SaddleUSDPool'	'0x8aCd85898458400f7Db866d53FCFF6f0D49741FF'
17	'SaddleUSDPoolLPToken'	'0xe4c278D321184BBFFB72e4e59e16a953b6863BEF'

```
       ✓ Emits SynthIndex event
       ✓ Succeeds with correct currencyKey
       ✓ Reverts when currencyKey do not match
       ✓ Reverts when given index is not a synth
    calcTokenToSynth
       ✓ Succeeds to calculate wBTC -> sUSD
       ✓ Succeeds to calculate wBTC -> sDEFI
    tokenToSynth
       ✓ Reverts when minAmount is not reached
       ✓ Succeeds to swap wBTC -> sUSD then settle it
       ✓ Succeeds to swap wBTC -> sDEFI then settle it
       ✓ Reverts when minAmount is not reached
    calcSynthToToken
       ✓ Succeeds to calculate sUSD -> tBTC
       ✓ Succeeds to calculate sDEFI -> tBTC
    synthToToken
       ✓ Reverts when minMediumSynthAmount is not reached
       ✓ Succeeds to swap sUSD -> sBTC -> tBTC
    calcTokenToToken
       ✓ Succeeds to calculate tBTC -> sBTC -> sUSD -> USDC
       ✓ Succeeds to calculate USDC -> sUSD -> sBTC -> WBTC
    tokenToToken
       ✓ Reverts when minMediumSynthAmount is not reached
      Initiate a cross asset swap: tBTC -> sBTC -> sUSD -> USDC
        completeToToken
          ✓ Succeeds with the full amount
          ✓ Succeeds with partial amounts
          ✓ Reverts when not reached minAmount
        withdraw
          ✓ Succeeds to withdraw the synth in full amount
          ✓ Succeeds to withdraw in partial amounts
          ✓ Reverts when trying to withdraw more than the synth balance
```

Solc version: 0.6.12		Optimizer enabled: true		Runs: 10000	Block limit: 9500000 gas	
Methods						
Contract	Method	Min	Max	Avg	# calls	usd (avg)
Allowlist	setPoolAccountLimit	-	-	44098	4	-
Allowlist	setPoolCap	-	-	44097	4	-
Allowlist	updateMerkleRoot	-	-	28761	3	-
Allowlist	verifyAddress	23109	46012	33592	28	-
Bridge	completeToSynth	210061	217548	213805	4	-
Bridge	completeToToken	335141	404236	347224	6	-
Bridge	setSynthIndex	96359	96371	96368	31	-
Bridge	synthToToken	-	-	1109719	1	-
Bridge	tokenToSynth	1210381	1284395	1247388	4	-
Bridge	tokenToToken	-	-	1225585	6	-
Bridge	withdraw	151901	216725	195109	3	-
ERC20	approve	44379	66895	54120	21	-
ERC20	transfer	23949	168560	71976	9	-
FlashLoanBorrowerExample	flashLoan	-	-	64796	3	-
GenericERC20	approve	28998	44358	44063	62	-
GenericERC20	mint	51664	66736	58638	93	-
GenericERC20	transfer	51023	51035	51032	4	-
LPToken	approve	25158	44358	29526	45	-
LPToken	transfer	62059	104737	83398	2	-
LPToken	transferFrom	-	-	106260	1	-
StakeableTokenWrapper	stake	46008	76008	71722	7	-
StakeableTokenWrapper	withdraw	-	-	33827	2	-
Swap	addLiquidity	199698	534628	278620	98	-
Swap	initialize	1548026	1678305	1613178	4	-
Swap	pause	-	-	44992	5	-
Swap	rampA	75293	76079	75502	38	-
Swap	removeLiquidity	71038	169886	102181	20	-
Swap	removeLiquidityImbalance	189639	223228	207785	8	-
Swap	removeLiquidityOneToken	160930	168639	164422	6	-
Swap	setAdminFee	45428	45440	45433	7	-

```
|  Swap                     ·  setDefaultWithdrawFee          ·      15368  ·      45416  ·        41661  ·          24  ·          -  |
|  Swap                     ·  setSwapFee                     ·          -  ·          -  ·        30451  ·           3  ·          -  |
|  Swap                     ·  stopRampA                      ·          -  ·          -  ·        50146  ·           2  ·          -  |
|  Swap                     ·  swap                           ·     145049  ·     178589  ·       163023  ·          83  ·          -  |
|  Swap                     ·  unpause                        ·          -  ·          -  ·        15005  ·           1  ·          -  |
|  Swap                     ·  withdrawAdminFees              ·      40048  ·      74212  ·        62824  ·           3  ·          -  |
|  SwapDeployer             ·  deploy                         ·    1713760  ·    1737449  ·      1718503  ·           5  ·          -  |
|  SwapFlashLoan            ·  addLiquidity                   ·          -  ·          -  ·       454095  ·           1  ·          -  |
|  SwapFlashLoan            ·  initialize                     ·          -  ·          -  ·      1720181  ·           1  ·          -  |
|  SwapFlashLoan            ·  setFlashLoanFees               ·          -  ·          -  ·        32670  ·           1  ·          -  |
|  SwapGuarded              ·  disableGuard                   ·          -  ·          -  ·        14033  ·           4  ·          -  |
|  TestSwapReturnValues     ·  test_addLiquidity              ·          -  ·          -  ·       300724  ·           5  ·          -  |
|  TestSwapReturnValues     ·  test_removeLiquidity           ·          -  ·          -  ·       169982  ·           1  ·          -  |
|  TestSwapReturnValues     ·  test_removeLiquidityImbalance  ·          -  ·          -  ·       234593  ·           1  ·          -  |
|  TestSwapReturnValues     ·  test_removeLiquidityOneToken   ·          -  ·          -  ·       226557  ·           1  ·          -  |
|  TestSwapReturnValues     ·  test_swap                      ·          -  ·          -  ·       183726  ·           1  ·          -  |
|  Deployments                                                ·                                          ·  % of limit  ·             |
|  Allowlist                                                  ·          -  ·          -  ·       815750  ·       8.6 %  ·          -  |
|  Bridge                                                     ·          -  ·          -  ·      5534221  ·      58.3 %  ·          -  |
|  FlashLoanBorrowerExample                                   ·          -  ·          -  ·       653938  ·       6.9 %  ·          -  |
|  GenericERC20                                               ·    1072566  ·    1072722  ·      1072619  ·      11.3 %  ·          -  |
|  LPToken                                                    ·          -  ·          -  ·      1300284  ·      13.7 %  ·          -  |
|  MathUtils                                                  ·          -  ·          -  ·       114689  ·       1.2 %  ·          -  |
|  StakeableTokenWrapper                                      ·          -  ·          -  ·       578238  ·       6.1 %  ·          -  |
|  Swap                                                       ·    4394640  ·    4394904  ·      4394902  ·      46.3 %  ·          -  |
|  SwapDeployer                                               ·          -  ·          -  ·       650763  ·       6.9 %  ·          -  |
|  SwapFlashLoan                                              ·          -  ·          -  ·      5102072  ·      53.7 %  ·          -  |
|  SwapGuarded                                                ·          -  ·          -  ·      4172840  ·      43.9 %  ·          -  |
|  SwapUtils                                                  ·          -  ·          -  ·      4044110  ·      42.6 %  ·          -  |
|  SwapUtilsGuarded                                           ·          -  ·          -  ·      4213392  ·      44.4 %  ·          -  |
|  TestSwapReturnValues                                       ·          -  ·          -  ·      1701825  ·      17.9 %  ·          -  |

  225 passing (3m)
```

Code Coverage

The code coverage scripts do not currently handle the VirtualSwap contracts due to a mainnet dependency.

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

```
e38ac7efbdcf842db9d90c3fea11e98c9661e14b5e99744be2fabd34fd765f10 ./contracts/SwapFlashLoan.sol
aa5a611727bc5fa1ee6e3286182846909eeaed8a0a58dd15840927271ae415e3 ./contracts/MathUtils.sol
f4d79582b67df36d82648a8b7af60b14850e46713c5886acd2350f07677b0a28 ./contracts/SwapDeployer.sol
c44affe35f4a86bcfc1c3e55330ecab8c7d279aceb37a4f7ff67810f9a19aa81 ./contracts/OwnerPausableUpgradeable.sol
fab569ac64d66193f05f7de14d021c57751a1a6ed8eee3493589dc65a684825a ./contracts/StakeableTokenWrapper.sol
8616297461c3dfc01f214f0eea9170e3157d089254c1008b3b117b9911b41854 ./contracts/LPToken.sol
3aab4d2301bf13631af13d2f9125afd9d129b36379d47570508496831d6d79d8 ./contracts/Swap.sol
9812c5a9c2e8a242cad31e4739e6641ea10566af47398ab5542a75b17f57d5b2 ./contracts/SwapUtils.sol
7b96876fc12505f74729a70aaa63e8cf78742474837f97b2b32604842c8761d4 ./contracts/interfaces/IFlashLoanReceiver.sol
7ff1e26341789a794ee6542cc9507b0200225d3797f96b478ad4a2ca4bf4e8e4 ./contracts/interfaces/IAllowlist.sol
22ee52c76ac29947b7a510086ac31b270c4d5ca80e20018cff30e771e02a7342 ./contracts/interfaces/ISwap.sol
9dae47c485ff6988fc457389ff7fe8a1de8d2de0ce285b5ecbfd55217e700ae3 ./contracts/interfaces/ISwapFlashLoan.sol
077ec0265ad29bf6133fd5181219938829932e4e4296f1c868d1db5b99b1bf8e ./contracts/interfaces/ISwapGuarded.sol
764ddb44a85a4a852974cabc654d7801fa4f0f577af4c0f0b67ff07b9ad79709 ./contracts/helper/FlashLoanBorrowerExample.sol
0f6f94dd4e8b44b36d18b6e56804a28b8e3719cc0d6041c69aacabd6a254d586 ./contracts/helper/GenericERC20.sol
5187d232f17e9d54930a79c71703f4cccd2c38a3cfee2a590ea05891e390490e ./contracts/helper/test/TestSwapReturnValues.sol
a155b4381e1a6b5801e616bbfc4b5696f77cf55bead2f98951651f84638ca415 ./contracts/VirtualSwap/Bridge.sol
11531f51926013e586e9832b6865333c8ac1e69ded363d72afcca39a342f177b ./contracts/VirtualSwap/SynthSwapper.sol
1a695d67c0222c095ca5d7f83d1f025e0378dc0504afe21cbeea1e77403e2c9f ./contracts/guarded/OwnerPausable.sol
79836cd81df43f5a3891db79f315496dee6b05f345353f0b06438c320d00cdab ./contracts/guarded/SwapUtilsGuarded.sol
ffe4c2282fe44a009faa0c577d2e2caa8f1000b5cf4b88d2c94bbcc39f98b355 ./contracts/guarded/Allowlist.sol
30b25e55fc8686bee317e23824f11f6c0f3b3503687bf000fc39470bc973bb48 ./contracts/guarded/SwapGuarded.sol
005cfd5763bcdf308ca7a2674f790e43ed2f9be3405d25c03d499057c56a3026 ./contracts/guarded/LPTokenGuarded.sol
```

Tests

```
6f157630b4ccd0279a0887b3e9bd62e72e8f2dec8c1e640824d24fdab4882c0f ./test/lpToken.ts
8dd00cadbb4d99ea3be0ecbe062575e8c0c871726aa4edd4d592cf4758486d55 ./test/virtualBridge.ts
be4cac1ab2bb9ba97ee0863993fe92fc0d64686e362231ec2f63ff718a0de428 ./test/mathUtils.ts
ac81a3ae6ae442bfe3e0797c3dae8a823fa2aadf56ded535d23ff2056ada64d1 ./test/swapInitialize.ts
0949108ab466b3b02385aae84ebbe0ce6df4dfa1fd01588e5d39543611e73e3b ./test/allowlist.ts
02a5d923537e1a467e8f6fa36454730e587356bfcc55f31239ec772343fb8e0f ./test/genericERC20.ts
a77b97808fa8cc34d3c2baa157fd1ba9414f89f0a21ae84206c4663b27006b7c ./test/flashloan.ts
dec49dd70632839d1378b227a1a24e4758a59e6aa2d830a39de11effe8d5c62b ./test/swapDeployer.ts
d5fcbc87915c8b24ee245734331addf0d9d35f63c4df8e2a573f83cb11b8639a ./test/testUtils.ts
fccf2e590957110c8f249ace44adfac7b35853ae16644ec3f0d7698b1c182da9 ./test/swap.ts
8e57a0121443812df3e8f5739fb450c134ce0cfe302c440db19f00d4c6aca0ea ./test/stakeableTokenWrapper.ts
dbc1c37cc026bb74351fb554e6cfbf2ef8aa7bc977f4f500a420f1eac623cefa ./test/swap4tokens.ts
```

Changelog

- 2021-03-07 - Initial report
- 2021-03-29 - Updated report based on commit b0c731ce

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.