Quantstamp Security Assessment Certificate

October 5th 2021 — Quantstamp Verified

Popsicle

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

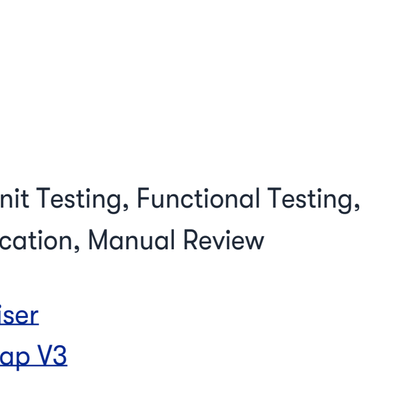
Executive Summary

Type	Yield Optimizer
Reviewers	Fayçal Lalidji, Security Auditor Jose Ignacio Orlicki, Senior Engineer
Timeline	2021-09-06 through 2021-10-04
EVM	Muir Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Popsicle Uni V3 optimiser Interaction with Uniswap V3 Call Graph
Documentation Quality	High
Test Quality	Low

Source Code

Repository	Commit
PopsicleV3Optimizer	d151377
None	540b63f

Total Issues	9 (5 Resolved)
High Risk Issues	2 (2 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	2 (1 Resolved)
Informational Risk Issues	5 (2 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

Initial Audit

Through reviewing the code, we found **9 potential issues** of various levels of severity. We recommend addressing the findings prior to deploying the smart contracts to the main network.

Re-audit Update

All highlighted issues were either fixed or acknowledged.

ID	Description	Severity	Status
QSP-1	Possible Truncation While Calculating Users Deposits Shares	High	Mitigated
QSP-2	Un-compounded Protocol Fees	High	Fixed
QSP-3	Input Validation	Low	Acknowledged
QSP-4	Possible Incorrect Execution Order in `collectProtocolFees()`	Low	Fixed
QSP-5	MEV Protection Can Disrupt Gas Fees	Informational	Fixed
QSP-6	Unchecked Transfers	Informational	Acknowledged
QSP-7	Events Missing For State Changes	Informational	Acknowledged
QSP-8	Unlocked Pragma	Informational	Mitigated
QSP-9	Clone-and-Own	Informational	Acknowledged

Quantstamp Review 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 reviewing 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 security review.

Setup

Tool Setup:

- Slither V0.8.0

Steps taken to run the tools:

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

Findings

QSP-1 Possible Truncation While Calculating Users Deposits Shares

Severity: *High Risk*

Status: Mitigated

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: The implemented staking mechanism in `PopsicleV3Optimizer` is highly optimized and helps avoid any extra computation. However, setting the values of the initial shares for a specific pool is important and should be analyzed carefully.
The algorithm is designed in a way to compute smaller shares every time that the pool generates fees, meaning that depositing the same liquidity value will result in giving different shares values (while the fees are accruing).
However, if the initial liquidity deposit is low enough and if the fees generated later on are significantly greater than the initial deposit, the new depositors to the optimizer will possibly see their computed share being truncated.
In a worst-case scenario, if for any reason higher fees can be introduced by an attacker (which can be done by directly transferring one of the pair assets to the pool), the next deposits will be truncated following the value of fees introduced. The attacker will profit by getting higher returns since users' shares might be truncated following the deposited liquidity.

Recommendation: This issue can be solved by multiplying the initial liquidity by a constant multiplier in `_calcShare()` in the case if `totalSupply()` equal zero. The value of the multiplier should be determined following the smallest possible investment, for example, if the smallest possible investment is 1 Wei, we would recommend a multiplier of `10**18`.

Update: The multiplier value was reduced to a different magnitude (1e6) by the project team.

QSP-2 Un-compounded Protocol Fees

Severity: *High Risk*

Status: Fixed

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: In the `withdraw()` function, protocol fees are updated without compounding them, meaning that when the protocol calculated liquidity is subtracted from the total pool liquidity the newly earned fees are not accounted for in the pool total liquidity (not compounded initially), this will reduce what the users' is supposed to receive when burning his shares.

Recommendation: It is not described why `_compoundFees()` is not called right after `_earnFees()` is executed. This possible fix will avoid extra gas consumption since the generated users' fees will also be compounded, meaning that calculating them separately will be unnecessary.

QSP-3 Input Validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: It is rarely desirable for a token to use the `0x0` address (although intentional token burning is a notable exception). However, these mistakes are often made due to human errors. Hence, it is usually a good idea to prevent these mistakes from happening within the smart contract by adding validation to check for `0x0` address inputs. Currently, checks for `0x0` address are missing in the following: * `_strategy` in `PopsicleV3Optimizer.constructor()` * `_governance` in `PopsicleV3Optimizer.setGovernance()` * `to` in `PopsicleV3Optimizer.deposit()`

Recommendation: Adding a `require` statement to check for the argument to be different `0x0` in the stated input parameters above is recommended.

Update: Team answers:

- "If we set strategy to 0x0 call to init of the optimizer will fail if optimizer will not be inited other methods will not work. "
- " Default state is 0x0 zero address can not accept it anyway "

QSP-4 Possible Incorrect Execution Order in `collectProtocolFees()`

Severity: *Low Risk*

Status: Fixed

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: In `collectProtocolFees()`, `_earnFees()` is executed after the requirements that validate the amount to be withdrawn by the governance. However, if `_earnFees()` is executed before the values of the protocol fees variables might increase following the fees earned by the pool.

Recommendation: Depending on the protocol design, the issue should be addressed by executing `_earnFees()` before the initial function requirements.

QSP-5 MEV Protection Can Disrupt Gas Fees

Severity: *Informational*

Status: Fixed

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: Given that the MEV protection in `rerange()` and `rebalance()` is like a bribe or extra reward for the miner, this can disrupt the gas fees incentives since London Fork in Ethereum main network. This occurs because since London Fork part of the transaction gas fees is burned, then miners will give more priority to `reragne()` and `rebalance()` transactions with bigger `msg.value` attached, to reduce the net fees burnt.

Recommendation: Remove these bribes or write better documentation on how they will enhance the MEV protection without damaging the Ethereum main network incentives.

QSP-6 Unchecked Transfers

Severity: *Informational*

Status: Acknowledged

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: Even if is not exploitable because it never fails, is a good practice to always check the return value of unsafe `transfer()` or `transferFrom()` ERC20s functions. In this case, in `pay()` the call to `IWETH9(weth).transfer(recipient, value)` is never checked.

Recommendation: Check the boolean value return and revert in case of negative results.

Update: Team answer: "We do not have enough space for good practice unfortunately, bytes will exceed maximum amount".

QSP-7 Events Missing For State Changes

Severity: *Informational*

Status: Acknowledged

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: IF the reconstruction of the state using the logs or at least registering all critical state changes is wanted, some events might be missing. For example, in `setStrategy()` an event should be emitted.

Recommendation: Declare and emit the necessary events correctly.

Update: Team answer: "We do not have enough space for good practice unfortunately, bytes will exceed maximum amount ".

QSP-8 Unlocked Pragma

Severity: *Informational*

Status: Mitigated

File(s) affected: `PopsicleV3Optimizer_flat.sol`

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.*.*`. The caret (^) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

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

Update: Compiler version is not locked in all contracts.

QSP-9 Clone-and-Own

Severity: *Informational*

Status: Acknowledged

File(s) affected: `PopsicleV3Optimizer_flat.sol`

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

Recommendation: Rather than the clone-and-own approach, a good industry practice is to use the hardhat framework for managing library dependencies. This eliminates the clone-and-own risks yet allows for following best practices, such as, using libraries.

Update: The team answer was "the code was cloned to be modified by the team".

Automated Analyses

Slither

Slither did not report any significant issues.

Test Results

Test Suite Results

```
npx hardhat test

Compiling 40 files with 0.7.6
Generating typings for: 72 artifacts in dir: typechain for target: ethers-v5
Successfully generated 52 typings!
Successfully generated 15 typings for external artifacts!
Compilation finished successfully

  OptimizerStrategy
    setMaxTotalSupply
       ✓ should set new maxTotalSupply (45ms)
       ✓ should execute only by the owner
       ✓ should check input value
    setTwapDuration
       ✓ should set new twapDuration
       ✓ should execute only by the owner
       ✓ should check input value
    setMaxTwapDeviation
       ✓ should set new maxTwapDeviation
       ✓ should execute only by the owner
       ✓ should check input value
    setTickRange
       ✓ should set new tickRangeMultiplier
       ✓ should execute only by the owner
    setPriceImpact
```

```
      ✓ should set new priceImpactPercentage
      ✓ should execute only by the owner
      ✓ should check input value
    setGovernance
      ✓ should set new pendingGovernance
      ✓ should execute only by the owner
    acceptGovernance
      ✓ should accept new governance
      ✓ should execute only by the pending governance

  PopsicleV3Optimizer
    init
      ✓ should init contract
      ✓ should execute only by the owner
      ✓ fails if already initialized
    deposit
      ✓ should emit Deposit event (162ms)
      ✓ should check for zero amount (40ms)
      ✓ should check for don't paused
    withdraw
      ✓ should emit Withdraw event (90ms)
      ✓ should check for shares amount
      ✓ should check for address(0)
      ✓ should check for don't paused (43ms)
    rerange
      ✓ should emit Rerange event (137ms)
      ✓ should call only operator
    rebalance
      ✓ should emit Rerange event (227ms)
      ✓ should call only operator (38ms)
    position
      ✓ should return correct position
    uniswapV3MintCallback
      ✓ should call only pool
    uniswapV3SwapCallback
      ✓ should call only pool
    collectProtocolFees
      ✓ should emit RewardPaid event
      ✓ should execute only by the owner
      ✓ should chack for protocolFees0 >= amount0
      ✓ should chack for protocolFees1 >= amount1
    setGovernance
      ✓ should set new pending governance
      ✓ should execute only by the owner
    acceptGovernance
      ✓ should emit event TransferGovernance
      ✓ should accept new governance
      ✓ should execute only by the pending governance
    setStrategy
      ✓ should change strategy
      ✓ should check for address(0)
      ✓ should execute only by the owner
    approveOperator
      ✓ should approved operator
      ✓ should execute only by the owner
    disableOperator
      ✓ should disable operator
      ✓ should execute only by the owner
    isOperator
      ✓ should check operator
    pause
      ✓ should paused
      ✓ should check for pause
      ✓ should execute only by the owner
    unpause
      ✓ should unpause
      ✓ should check for pause
      ✓ should execute only by the owner


  58 passing (22m)
```

Code Coverage

Quantstamp usually recommends developers to increase the branch coverage to 90%and above before a project goes live, in order to avoid hidden functional bugs that might not be easy to spot during the development phase. For code coverage, the current targeted files by the audit achieve poor scores that should be improved before live deployment, including a failing test.

```
npx hardhat coverage

Version
=======
> solidity-coverage: v0.7.17

Instrumenting for coverage...
=============================

> helpers/libraries/  LowGasSafeMath.sol
> helpers/token/  ERC20.sol
> helpers/token/  IERC20.sol
> helpers/utils/  Context.sol
> popsicle-v3-optimizer/base/  EIP712.sol
> popsicle-v3-optimizer/base/  ERC20Permit.sol
> popsicle-v3-optimizer/interfaces/external/  IWETH9.sol
> popsicle-v3-optimizer/interfaces/  IOptimizerStrategy.sol
> popsicle-v3-optimizer/interfaces/  IPopsicleV3Optimizer.sol
> popsicle-v3-optimizer/interfaces/  IUniswapV3Pool.sol
> popsicle-v3-optimizer/interfaces/pool/  IUniswapV3PoolActions.sol
> popsicle-v3-optimizer/interfaces/pool/  IUniswapV3PoolDerivedState.sol
> popsicle-v3-optimizer/interfaces/pool/  IUniswapV3PoolImmutables.sol
> popsicle-v3-optimizer/interfaces/pool/  IUniswapV3PoolState.sol
> popsicle-v3-optimizer/libraries/  Babylonian.sol
> popsicle-v3-optimizer/libraries/  ChainId.sol
> popsicle-v3-optimizer/libraries/  Counters.sol
> popsicle-v3-optimizer/libraries/  ECDSA.sol
> popsicle-v3-optimizer/libraries/  FixedPoint128.sol
> popsicle-v3-optimizer/libraries/  FixedPoint96.sol
> popsicle-v3-optimizer/libraries/  FullMath.sol
> popsicle-v3-optimizer/libraries/  LiquidityAmounts.sol
> popsicle-v3-optimizer/libraries/  LowGasSafeMath.sol
> popsicle-v3-optimizer/libraries/  PoolActions.sol
> popsicle-v3-optimizer/libraries/  PoolVariables.sol
> popsicle-v3-optimizer/libraries/  PositionKey.sol
> popsicle-v3-optimizer/libraries/  SafeCast.sol
> popsicle-v3-optimizer/libraries/  SqrtPriceMath.sol
> popsicle-v3-optimizer/libraries/  TickMath.sol
> popsicle-v3-optimizer/libraries/  TransferHelper.sol
> popsicle-v3-optimizer/libraries/  UnsafeMath.sol
> popsicle-v3-optimizer/  OptimizerStrategy.sol
> popsicle-v3-optimizer/  PopsicleV3Optimizer.sol
> popsicle-v3-optimizer/token/  ERC20.sol
> popsicle-v3-optimizer/token/  IERC20.sol
> popsicle-v3-optimizer/token/  IERC20Permit.sol
> popsicle-v3-optimizer/utils/  Context.sol
> popsicle-v3-optimizer/utils/  ReentrancyGuard.sol

Coverage skipped for:
=====================

> popsicle-v3-optimizer/  OptimizerStrategy_flat.sol
> popsicle-v3-optimizer/  PopsicleV3Optimizer_flat.sol

Compilation:
```

```
============

Nothing to compile
Generating typings for: 0 artifacts in dir: typechain for target: ethers-v5
Successfully generated 3 typings!
Successfully generated 15 typings for external artifacts!

Network Info
============
> HardhatEVM: v2.6.0
> network:    hardhat

Generating typings for: 0 artifacts in dir: typechain for target: ethers-v5
Successfully generated 3 typings!
Successfully generated 15 typings for external artifacts!


  OptimizerStrategy
    setMaxTotalSupply
       ✓ should set new maxTotalSupply
       ✓ should execute only by the owner
       ✓ should check input value
    setTwapDuration
       ✓ should set new twapDuration
       ✓ should execute only by the owner
       ✓ should check input value
    setMaxTwapDeviation
      1) should set new maxTwapDeviation
       ✓ should execute only by the owner
       ✓ should check input value
    setTickRange
       ✓ should set new tickRangeMultiplier (38ms)
       ✓ should execute only by the owner
    setPriceImpact
       ✓ should set new priceImpactPercentage
       ✓ should execute only by the owner
       ✓ should check input value
    setGovernance
       ✓ should set new pendingGovernance
       ✓ should execute only by the owner
    acceptGovernance
       ✓ should accept new governance
       ✓ should execute only by the pending governance

  PopsicleV3Optimizer
    init
       ✓ should init contract (73ms)
       ✓ should execute only by the owner (50ms)
       ✓ fails if already initialized (47ms)
    deposit
       ✓ should emit Deposit event (326ms)
       ✓ should check for zero amount (72ms)
       ✓ should check for don't paused (64ms)
    withdraw
       ✓ should emit Withdraw event (171ms)
       ✓ should check for shares amount (53ms)
       ✓ should check for address(0) (42ms)
       ✓ should check for don't paused (67ms)
    rerange
       ✓ should emit Rerange event (254ms)
       ✓ should call only operator (44ms)
    rebalance
       ✓ should emit Rerange event (419ms)
       ✓ should call only operator (44ms)
    position
       ✓ should return correct position
    uniswapV3MintCallback
       ✓ should call only pool
    uniswapV3SwapCallback
       ✓ should call only pool
    collectProtocolFees
       ✓ should emit RewardPaid event (56ms)
       ✓ should execute only by the owner
       ✓ should chack for protocolFees0 >= amount0
       ✓ should chack for protocolFees1 >= amount1
    setGovernance
       ✓ should set new pending governance
       ✓ should execute only by the owner
    acceptGovernance
       ✓ should emit event TransferGovernance
       ✓ should accept new governance
       ✓ should execute only by the pending governance
    setStrategy
       ✓ should change strategy
       ✓ should check for address(0)
       ✓ should execute only by the owner
    approveOperator
       ✓ should approved operator
       ✓ should execute only by the owner
    disableOperator
       ✓ should disable operator
       ✓ should execute only by the owner
    isOperator
       ✓ should check operator
    pause
       ✓ should paused
       ✓ should check for pause
       ✓ should execute only by the owner
    unpause
       ✓ should unpause
       ✓ should check for pause
       ✓ should execute only by the owner


57 passing (22m)
1 failing

1) OptimizerStrategy
     setMaxTwapDeviation
       should set new maxTwapDeviation:
   Error: VM Exception while processing transaction: reverted with reason string 'PF'
    at OptimizerStrategy.setMaxTwapDeviation (contracts/popsicle-v3-optimizer/  OptimizerStrategy.sol:128)
    at processTicksAndRejections (internal/process/task_queues.js:97:5)
    at runNextTicks (internal/process/task_queues.js:66:3)
    at listOnTimeout (internal/timers.js:523:9)
    at processTimers (internal/timers.js:497:7)
    at async HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1582:23)
    at async HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:442:16)
    at async EthModule._sendTransactionAndReturnHash (node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1500:18)
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
helpers/libraries/	16.67	8.33	16.67	16.67	
LowGasSafeMath.sol	16.67	8.33	16.67	16.67	… 6,84,92,100
helpers/token/	64.29	41.67	50	64.29	
ERC20.sol	64.29	41.67	50	64.29	… 256,257,289
IERC20.sol	**100**	**100**	**100**	**100**	
helpers/utils/	50	100	50	33.33	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
Context.sol	50	100	50	33.33	21,22
popsicle-v3-optimizer/	0	0	0	0	
OptimizerStrategy.sol	0	0	0	0	… ,99,100,101
PopsicleV3Optimizer.sol	0	0	0	0	… 588,599,610
popsicle-v3-optimizer/base/	37.5	0	33.33	37.5	
EIP712.sol	69.23	0	50	69.23	66,67,69,101
ERC20Permit.sol	0	0	20	0	… 73,80,81,82
popsicle-v3-optimizer/interfaces/	100	100	100	100	
IOptimizerStrategy.sol	100	100	100	100	
IPopsicleV3Optimizer.sol	100	100	100	100	
IUniswapV3Pool.sol	100	100	100	100	
popsicle-v3-optimizer/interfaces/external/	100	100	100	100	
IWETH9.sol	100	100	100	100	
popsicle-v3-optimizer/interfaces/pool/	100	100	100	100	
IUniswapV3PoolActions.sol	100	100	100	100	
IUniswapV3PoolDerivedState.sol	100	100	100	100	
IUniswapV3PoolImmutables.sol	100	100	100	100	
IUniswapV3PoolState.sol	100	100	100	100	
popsicle-v3-optimizer/libraries/	60.63	43.68	74.55	65.49	
Babylonian.sol	0	0	0	0	… 48,49,50,51
ChainId.sol	100	100	100	100	
Counters.sol	0	100	0	0	29,34
ECDSA.sol	0	0	0	0	… 30,31,33,46
FixedPoint128.sol	100	100	100	100	
FixedPoint96.sol	100	100	100	100	
FullMath.sol	38.1	40	100	37.93	… ,96,104,105
LiquidityAmounts.sol	67.74	50	100	82.61	66,73,129,134
LowGasSafeMath.sol	58.33	29.17	58.33	58.33	44,52,60,76,84
PoolActions.sol	68.75	33.33	66.67	68.75	74,75,76,78,80
PoolVariables.sol	89.36	58.33	91.67	89.36	88,90,93,94,95
PositionKey.sol	100	100	100	100	
SafeCast.sol	40	25	50	40	25,32,33
SqrtPriceMath.sol	55	35.71	100	57.89	… 44,47,77,84
TickMath.sol	83.93	73.91	100	100	
TransferHelper.sol	100	50	100	100	
UnsafeMath.sol	100	100	100	100	
popsicle-v3-optimizer/token/	0	0	0	0	
ERC20.sol	0	0	0	0	… 276,277,288
IERC20.sol	100	100	100	100	
IERC20Permit.sol	100	100	100	100	
popsicle-v3-optimizer/utils/	0	0	0	0	
Context.sol	0	100	0	0	17,21,22
ReentrancyGuard.sol	0	0	0	0	39,51,54,56,60
All files	**35.03**	**27.3**	**36.13**	**37.32**	

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

```
ade61a2a7da188b4e26138fa395c243d132e25bac866a0c7f1ea555f848ff578 ./popsicle-v3-optimizer/PopsicleV3Optimizer_flat.sol
```

Tests

```
94977985097c580bc0d534155c03206824276ef23b54025549c88ccd44a7380c ./test/OptimizerStrategy.spec.ts
aaf9e90384f8be9237b3ef5358b5696eac3f329de311f98e18e7764f96a3633e ./test/PopsicleV3Optimizer.spec.ts
e7eb52f0765c1275692279ca4fd4e6e37095e96792d100e2db8c320662ffa25b ./test/shared/random-number.ts
e0974253303c0b754020bee31322439af963f797a5a0e8745c0039e0e80b0db5 ./test/shared/index.ts
ead7183b5ad75f9c160a486952603cf035dbd8353e02e97733d8ce3e5d41b5a4 ./test/shared/expect.ts
73e020330fb40d2167883c1c0d6d3feee7c38bff9e6aa0776422c85db12de119 ./test/shared/constants.ts
284de79fecf21dcced4d3b2a079531feaeb66e8a3bc3705124262893cffd651a ./test/shared/popsicle/index.ts
f12537b069ccac793d9b0babd399a7372c65987e6729926670561e1b87f95117 ./test/shared/popsicle/liquidity/liquidity.ts
a46a04940a98c25c9e162f08d0690a19121c283d795f94870fed442547e7d8c8 ./test/shared/popsicle/liquidity/index.ts
0bfe3683306241eecbfdc2b133ac198520c2bcc72827fe3e068d37b18e3c197f ./test/shared/popsicle/liquidity/liquidity-for-amounts.ts
0880ef8d559dfafb984f0f93a03ab4c9cc3f4068ec7c9b0425033a5826f1fdb8 ./test/shared/popsicle/liquidity/liquidity-last.ts
7dc58dac5edfcdc6fc3e1735efa2dd866bbd68faeea1c96c5967d2fb7c9d92c3 ./test/shared/popsicle/helpers/get-ticks.ts
038b36e97cd3be844162acd990f6c0dd39a5c6277078d51eb21f9e97a781080d ./test/shared/popsicle/helpers/get-position-key.ts
648a71607526bf2b92d0fa55fc63d4e85e689c21785145132e709a3b6f7f69d5 ./test/shared/popsicle/helpers/index.ts
0de39885288a84d04211de4606794318c7cf4c0c80afbc0a7a25978fd4867ac7 ./test/shared/popsicle/helpers/pool-disbalancer.ts
5b1d5d79af3d05483432f41a294f3ff30ac51765d3d1d9d126017dece853560c ./test/shared/popsicle/helpers/calc-share.ts
027061c2a6760bbfdb6e8fbce3d7d74d9517d931e738a7c586e725b542aaaf1f ./test/shared/popsicle/helpers/deployers/index.ts
73a694195212bcc68255e11406b685666483a8ec60faf48bf8d0fb1afcb50436 ./test/shared/popsicle/helpers/deployers/deploy-strategy.ts
192e87945404988476ee41bed9732d7745d27ab4cf1b3ce7e97368807e13ea6f ./test/shared/uniswap/index.ts
89f1094268d4c583a3735a52b5316c92346b002c8f3a716996d5f4010e06972e ./test/shared/uniswap/pool/index.ts
3207da6074ca8be3ee78108fca015d150b7bbb39a0234c565b2d70a0077137f4 ./test/shared/uniswap/pool/burn-amounts.ts
6b2896333e4b1559d646ce87a4047a4b85b6a650e853685dd9b296ce854f6e9a ./test/shared/uniswap/pool/mint-amounts.ts
7f5bc78ae551d3061f4a651b602c4b418728498c8dad0fb8fda1737a2bb0c043 ./test/shared/uniswap/helpers/create-position.ts
def4954298c6c25961d1a0869893d015d212e54c5619338632eab1abc83159ba ./test/shared/uniswap/helpers/encode-price-sqrt.ts
ad06e6229b2ed4c405b2bc2c0ca6518c85903ea5a4932acd62c861ee1235939f ./test/shared/uniswap/helpers/index.ts
33d4e8d1fa06906c986967adc617b7dddeaffa6e344e33322ef049be364c2c53 ./test/shared/uniswap/helpers/deployers/deploy-uniswap-swap-router.ts
b0d077c9c65ff2ea7ebb01abb55dc2c64d62cd997e09552c334736b86e5bd5c8 ./test/shared/uniswap/helpers/deployers/deploy.ts
390867baa0338a07a6a16653f7e59c8004e49dda2175c850a852987e53eada84 ./test/shared/uniswap/helpers/deployers/index.ts
a7dff4a9c0594f06ff4919a9f96459e318ea355bc0ee4e6979fedf7dcb34bd13 ./test/shared/uniswap/helpers/deployers/deploy-weth9.ts
8f12b1b41e80a6b4f01d7e18ccfacf0992f8a9779ac4d846d4083b36e69ff639 ./test/shared/uniswap/helpers/deployers/deploy-uniswap-factory.ts
ecf26059dbc0877e6751f0c02e555c1c5cf48624a64d00854438d1d90a0a83bf ./test/shared/uniswap/helpers/deployers/deploy-erc20.ts
785856673c6e78fcf27f7e7db8950acb25061f19a76472fae33f81ae12593982 ./test/shared/uniswap/helpers/deployers/deploy-uniswap-pool.ts
533dad52fce1d6977a5b65d06ec639b9f65c8f13bbd1e51a7eb01f6bd8022cee ./test/shared/uniswap/constants/index.ts
2a385fddcf80feb39c1bd89ffff67b236c7ac01a816b28acf54df47efed4626f ./test/shared/uniswap/constants/fee-amount.enum.ts
d8d399d1412fd0086400a86e1531312871ddda23056e8ffa923ebb8eda14fdff ./test/shared/uniswap/router/index.ts
785f24dff7bc827931bef5d7371a0451292e8a167970ccf1eea0f9cd8c10c190 ./test/shared/uniswap/router/exact-input-single.ts
```

Changelog

- 2021-09-17 - Initial report
- 2021-10-04 - Re-audit report (540b63f)

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.