Quantstamp Security Assessment Certificate

September 20th 2021 — Quantstamp Verified

Tokemak

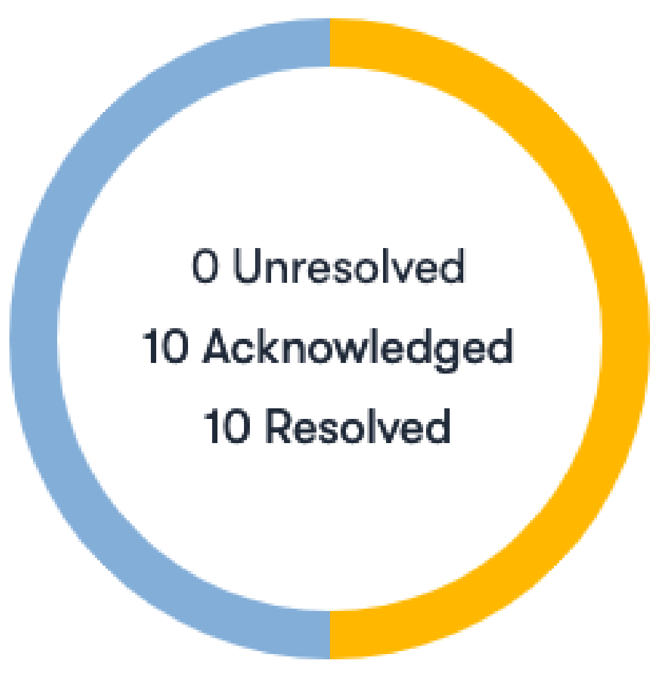
This audit report was prepared by Quantstamp, the leading blockchain security company.

Executive Summary

Type	DeFi - Liquidity Provider
Auditors	Mohsen Ahmadvand, Senior Research Engineer Poming Lee, Research Engineer Kacper Bąk, Senior Research Engineer
Timeline	2021-07-19 through 2021-08-31
EVM	Muir Glacier
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Medium
Source Code	

Repository	Commit
tokemak-smart-contracts	8f33ff8
tokemak-smart-contracts	1dc68ff (re-audit)

Total Issues	20	(10 Resolved)
High Risk Issues	2	(1 Resolved)
Medium Risk Issues	1	(0 Resolved)
Low Risk Issues	8	(4 Resolved)
Informational Risk Issues	8	(4 Resolved)
Undetermined Risk Issues	1	(1 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

In the security audit of the selected contracts of Tokemak (listed in the appendix) we identified two high-severity, one medium-severity, and several low/informational/undetermined-severity issues. One high-severity issue was fixed. A remaining high-severity problem is the assumed trust in the manager. That is, the manager can perform any operation with the provided user funds based upon some calculations that are carried out off the chain. The staking logic does not force locking funds on the contract side (to which we assigned a medium-severity label). The Tokemak team decided upon not fixing the locking issue. Nevertheless, they have resolved 10 issues and acknowledged all the remaining issues.

ID	Description	Severity	Status
QSP-1	`_executeControllerCommand` Can be Used to Deploy or Withdraw Liquidity by Anyone	High	Fixed
QSP-2	Manager's Decisions Are Made Off the Chain	High	Acknowledged
QSP-3	Stake Without Locking Funds	Medium	Acknowledged
QSP-4	Funds Held in `BalancerController` Could be Drained	Low	Acknowledged
QSP-5	`initialize` Functions Can be Frontrun	Low	Mitigated
QSP-6	Gas Usage / `for` Loop Concerns	Low	Acknowledged
QSP-7	Potential Precision Loss in the effectiveSecondsStaked	Low	Fixed
QSP-8	Missing Parameter Validation	Low	Fixed
QSP-9	Unlimited Allowance for External Contract	Low	Fixed
QSP-10	Allowance Double-Spend Exploit	Low	Acknowledged
QSP-11	Unlocked Pragma	Low	Acknowledged
QSP-12	Sandwich Attacks Susceptibility in Sushiswap and Uniswap Controllers	Informational	Acknowledged
QSP-13	Unsafe Cycle Durations Can be Set	Informational	Acknowledged
QSP-14	Some Functions do not Conform to Contract Pauses	Informational	Fixed
QSP-15	`schedule.setup` Checks Always Pass	Informational	Acknowledged
QSP-16	Schedules Cannot be Modified	Informational	Fixed
QSP-17	ZeroExTradeWallet is Built Upon High Trust in Router	Informational	Fixed
QSP-18	Sushiswap funds are not eligible for SUSHI reward	Informational	Fixed
QSP-19	Privileged Roles and Ownership	Informational	Acknowledged
QSP-20	ZeroExTradeWallet Accepts External Token Addresses and Triggers Untrusted Code	? 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 v0.8.0

Steps taken to run the tools:

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

Findings

QSP-1 `_executeControllerCommand` Can be Used to Deploy or Withdraw Liquidity by Anyone

Severity: High Risk

Status: Fixed

File(s) affected: `contracts/manager/Manager.sol`

Description: `_executeControllerCommand` is defined as a public function. This function carries a very dangerous logic, i.e., executing arbitrary transactions without any access control.

Recommendation: Define the `_executeControllerCommand` function as internal/private. it appears that the needed functions are deploy and withdraw funds, consider implementing the corresponding functions instead of having a shell-access function in the contract.

QSP-2 Manager's Decisions Are Made Off the Chain

Severity: High Risk

Status: Acknowledged

File(s) affected: `contracts/manager/Manager.sol`

Description: The core flow of the system (i.e., supplying liquidity) is currently carried out by the admin team in an off-chain fashion. Precisely put, the calculations of slashing, reward distributions, and asset allocations take place of the chain. Based upon those calculations, the manager submits the transactions to the blockchain. This design assumes a high trust in the manager. The manager can be compromised or go rogue in which case users can lose funds.

Recommendation: Consider moving calculations on the chain. If this is not possible, the risks need be explicitly stated in the public facing platform documentation. Measures such as multi-sig wallets can improve the situation and mitigate the risk of rogue insiders.

QSP-3 Stake Without Locking Funds

Severity: Medium Risk

Status: Acknowledged

File(s) affected: `contracts/staking/Staking.sol`

Description: An attacker can obtain staking power if there is any `schedule` with a passed `hardStart`. The code never checks if funds are being deposited to an already expired `schedule`. Moreover, the code does not unset the `isActive` flag for expired schedules. This could result in a malicious user staking a large amount of funds in order to vote for their favorable target controller followed with an immediate withdraw of their funds. This scenario could be used as a utility to conduct a more complex attack.

Recommendation: In function `_depositFor`, checks and sets the `isActive` flag to `false` if `schedule.hardStart > 0` and the `schedule` has expired.

Update: Tokemak's response: "Will not fix; Expected behavior for the 'special staking' Off chain computations would ignore this otherwise"

QSP-4 Funds Held in `BalancerController` Could be Drained

Severity: Low Risk

Status: Acknowledged

File(s) affected: `contracts\controllers\BalancerController.sol`

Description: The visibility of function `deploy` and `withdraw` are set to `external` and without any permission check. With this, an attacker can withdraw all the funds that are in the other pools first, then drain the funds returned to the contract by deploying funds into a malicious pool under the attacker's control. Right now, the manager uses the vulnerable controller through delegate calls. Therefore, the manager's storage is used for the platform funds. If someone directly calls the controller it would net no funds.

Recommendation: Having such strong assumption on the consumption of controllers reduces the readability and increases the chance of making mistakes. Consider adding relevant permission checks to the functions.

QSP-5 `initialize` Functions Can be Frontrun

Severity: Low Risk

Status: Mitigated

File(s) affected: `contracts/manager/Manager.sol`, `contracts/pools/EthPool.sol`, `contracts/staking/Staking.sol`

Description: The `initialize` function that initializes an important contract state can be called by anyone. The attacker can initialize the contract before the legitimate deployer, hoping that the victim continues to use the same contract. In the best case for the victim, they notice it and have to redeploy their contract costing them gas.

Recommendation: Use the constructor to initialize non-proxied contracts. For initializing proxy contracts deploy contracts using a factory contract that immediately calls init after deployment or make sure to call it immediately after deployment and verify the transaction succeeded.

Update: Tokemak response:

> We will be upgrading all of our scripts to use the 'upgradeAndCall' functionality. We also immediately call initialize in all of out deploy scripts, lessening the chance of anyone except possible a bot sneaking in and taking control of the contract.

QSP-6 Gas Usage / `for` Loop Concerns

Severity: Low Risk

Status: Acknowledged

File(s) affected: `contracts/manager/Manager.sol`, `contracts/staking/Staking.sol`,

Description: Some loops are not capped at a max number of supported iterations. This has certain implications on the gas usage.

> Gas usage is a main concern for smart contract developers and users, since high gas costs may prevent users from wanting to use the smart contract. Even worse, some gas usage issues may prevent the contract from providing services entirely. For example, if a `for` loop requires too much gas to exit, then it may prevent the contract from functioning correctly entirely. It is best to break such loops into individual functions as possible.

Listed are the loops that we deem as unsafe with respect to gas usage:

1. `contracts/manager/Manager.sol:executeMaintenance:L125` function:

```
for (uint256 x = 0; x < params.cycleSteps.length; x++)
```

1. `contracts/manager/Manager.sol:executeRollover:L147` function:

```
for (uint256 x = 0; x < params.cycleSteps.length; x++) {
            _executeControllerCommand(params.cycleSteps[x]);
}
```

1. `contracts/manager/Manager.sol:L152`

```
for (uint256 y = 0; y < params.poolsForWithdraw.length; y++)
```

1. contracts/staking/Staking.sol:102~104:

```
 for(uint256 i = 0; i < length; i++) {
            retSchedules[i] = StakingScheduleInfo(schedules[scheduleIdxs.at(i)], scheduleIdxs.at(i));
 }
```

Recommendation: Ensure that loops are bound to a safe-max number of iterations. The number of active pools need to be capped with a max value to keep the gas price reasonable and to conform with the block size limits. Add a check to limit the number of registered pools in the manager contract.

QSP-7 Potential Precision Loss in the effectiveSecondsStaked

Severity: Low Risk

Status: Fixed

File(s) affected: `contracts/staking/Staking.sol:321`

Description: Division takes place before multipliciation:

```
uint256 effectiveSecondsStaked = (secondsStaked.div(schedule.interval)).mul(schedule.interval);
```

This potentially leads to precision loss.

Recommendation: Consider multiplying before dividing in the statement:

```
uint256 effectiveSecondsStaked = (secondsStaked.mul(schedule.interval)).div(schedule.interval);
```

QSP-8 Missing Parameter Validation

Severity: Low Risk

Status: Fixed

File(s) affected: `ZeroExTradeWallet.sol`, `SushiswapController.sol`

Description: The listed parameters are not validated properly:

1. `contracts/wallet/ZeroExTradeWallet.sol:32~33`: check newRouter and newManager against zero addresses
2. `contracts/wallet/ZeroExTradeWallet.sol:46~47`: check `signer` against zero
3. `contracts/controllers/SushiswapController.sol:25~26`: check router snd factory against zero
4. `contracts\manager\Manager.sol::_executeControllerCommand`, should check if `registeredControllers[transfer.controllerId]` is not zero.
5. `contracts\controllers\ZeroExController.sol::constructor`, should check if the input addresses are non-zeros.
6. `contracts\controllers\UniswapController.sol::constructor`, should check if the input addresses are non-zeros.
7. `contracts\pools\Pool.sol::initialize`, should check if the input addresses are non-zeros.
8. `contracts\pools\EthPool.sol::initialize`, should check if the input addresses are non-zeros.

Recommendation: Validate the parameters.

QSP-9 Unlimited Allowance for External Contract

Severity: Low Risk

Status: Fixed

Description: By calling the functions listed below, the allowance is set to a very large number. This could be used by external contracts that are not under admin team's control to drain the contract, whenever the target becomes malicious, or being controlled by malicious users.

1. `contracts\controllers\SushiswapController.sol::_approve`.
2. `contracts\controllers\ZeroExController.sol::_approve`.
3. `contracts\controllers\BalancerController.sol::_approve`.
4. `contracts\controllers\UniswapController.sol::_approve`.
5. `contracts\controllers\ZeroExTradeWallet.sol::_approve`.

Recommendation: Set up the allowance only when it is necessary, and never make it unlimited.

QSP-10 Allowance Double-Spend Exploit

Severity: Low Risk

Status: Acknowledged

File(s) affected: `contracts/token/Toke.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

QSP-11 Unlocked Pragma

Severity: Low Risk

Status: Acknowledged

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.4.*`. 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".
Different versions of Solidity is used:

```
- Version used: ['0.6.11', '>=0.4.24<0.8.0', '>=0.5.0', '>=0.6.0', '>=0.6.0<0.8.0', '>=0.6.2', '>=0.6.2<0.8.0', '^0.6.0', '^0.6.5']
- ^0.6.5 (node_modules/@0x/contracts-erc20/contracts/src/v06/IERC20TokenV06.sol#20)
- ^0.6.5 (node_modules/@0x/contracts-utils/contracts/src/v06/LibSafeMathV06.sol#20)
- ^0.6.5 (node_modules/@0x/contracts-utils/contracts/src/v06/errors/LibRichErrorsV06.sol#20)
- ^0.6.5 (node_modules/@0x/contracts-utils/contracts/src/v06/errors/LibSafeMathRichErrorsV06.sol#20)
- ^0.6.5 (node_modules/@0x/contracts-zero-ex/contracts/src/errors/LibNativeOrdersRichErrors.sol#20)
- ^0.6.5 (node_modules/@0x/contracts-zero-ex/contracts/src/errors/LibSignatureRichErrors.sol#20)
- ^0.6.5 (node_modules/@0x/contracts-zero-ex/contracts/src/features/interfaces/INativeOrdersEvents.sol#20)
- ABIEncoderV2 (node_modules/@0x/contracts-zero-ex/contracts/src/features/interfaces/INativeOrdersEvents.sol#21)
- ^0.6.5 (node_modules/@0x/contracts-zero-ex/contracts/src/features/interfaces/INativeOrdersFeature.sol#20)
- ABIEncoderV2 (node_modules/@0x/contracts-zero-ex/contracts/src/features/interfaces/INativeOrdersFeature.sol#21)
- ^0.6.5 (node_modules/@0x/contracts-zero-ex/contracts/src/features/libs/LibNativeOrder.sol#20)
- ABIEncoderV2 (node_modules/@0x/contracts-zero-ex/contracts/src/features/libs/LibNativeOrder.sol#21)
- ^0.6.5 (node_modules/@0x/contracts-zero-ex/contracts/src/features/libs/LibSignature.sol#20)
- ABIEncoderV2 (node_modules/@0x/contracts-zero-ex/contracts/src/features/libs/LibSignature.sol#21)
- >=0.6.0 (node_modules/@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol#2)
- ^0.6.0 (node_modules/@gnosis.pm/mock-contract/contracts/MockContract.sol#1)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/AccessControlUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/math/MathUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/math/SafeMathUpgradeable.sol#3)
- >=0.4.24<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/proxy/Initializable.sol#4)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/token/ERC20/SafeERC20Upgradeable.sol#3)
- >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/EnumerableSetUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts-upgradeable/utils/PausableUpgradeable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/AccessControl.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/access/Ownable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/cryptography/ECDSA.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/cryptography/MerkleProof.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/Math.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/math/SafeMath.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/presets/ERC20PresetMinterPauser.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20Burnable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20Pausable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/SafeERC20.sol#3)
- >=0.6.2<0.8.0 (node_modules/@openzeppelin/contracts/utils/Address.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/EnumerableSet.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/Pausable.sol#3)
- >=0.6.0<0.8.0 (node_modules/@openzeppelin/contracts/utils/SafeCast.sol#3)
- >=0.5.0 (node_modules/@sushiswap/core/contracts/uniswapv2/interfaces/IUniswapV2ERC20.sol#3)
- >=0.5.0 (node_modules/@sushiswap/core/contracts/uniswapv2/interfaces/IUniswapV2Factory.sol#3)
- >=0.6.2 (node_modules/@sushiswap/core/contracts/uniswapv2/interfaces/IUniswapV2Router01.sol#3)
- >=0.6.2 (node_modules/@sushiswap/core/contracts/uniswapv2/interfaces/IUniswapV2Router02.sol#3)
- >=0.5.0 (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2ERC20.sol#1)
- >=0.5.0 (node_modules/@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol#1)
- >=0.6.2 (node_modules/@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router01.sol#1)
- >=0.6.2 (node_modules/@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol#1)
- 0.6.11 (contracts/Imports.sol#3)
- 0.6.11 (contracts/airdrop/AirdropPush.sol#2)
- 0.6.11 (contracts/controllers/BalancerController.sol#3)
- ABIEncoderV2 (contracts/controllers/BalancerController.sol#4)
- 0.6.11 (contracts/controllers/SushiswapController.sol#3)
- ABIEncoderV2 (contracts/controllers/SushiswapController.sol#4)
- 0.6.11 (contracts/controllers/UniswapController.sol#3)
- ABIEncoderV2 (contracts/controllers/UniswapController.sol#4)
- 0.6.11 (contracts/controllers/ZeroExController.sol#3)
- ABIEncoderV2 (contracts/controllers/ZeroExController.sol#4)
```

```
    - 0.6.11 (contracts/core/CoreEvent.sol#2)
    - ABIEncoderV2 (contracts/core/CoreEvent.sol#3)
    - 0.6.11 (contracts/defi-round/DefiRound.sol#3)
    - ABIEncoderV2 (contracts/defi-round/DefiRound.sol#4)
    - 0.6.11 (contracts/interfaces/ICoreEvent.sol#2)
    - ABIEncoderV2 (contracts/interfaces/ICoreEvent.sol#3)
    - 0.6.11 (contracts/interfaces/IDefiRound.sol#3)
    - ABIEncoderV2 (contracts/interfaces/IDefiRound.sol#4)
    - 0.6.11 (contracts/interfaces/ILiquidityEthPool.sol#3)
    - 0.6.11 (contracts/interfaces/ILiquidityPool.sol#3)
    - 0.6.11 (contracts/interfaces/IManager.sol#3)
    - ABIEncoderV2 (contracts/interfaces/IManager.sol#4)
    - 0.6.11 (contracts/interfaces/IStaking.sol#3)
    - ABIEncoderV2 (contracts/interfaces/IStaking.sol#4)
    - 0.6.11 (contracts/interfaces/IWETH.sol#3)
    - 0.6.11 (contracts/interfaces/IWallet.sol#3)
    - 0.6.11 (contracts/interfaces/balancer/IBalancerPool.sol#3)
    - 0.6.11 (contracts/manager/Manager.sol#3)
    - ABIEncoderV2 (contracts/manager/Manager.sol#4)
    - 0.6.11 (contracts/pools/EthPool.sol#3)
    - 0.6.11 (contracts/pools/Pool.sol#3)
    - 0.6.11 (contracts/redeem/Redeem.sol#3)
    - 0.6.11 (contracts/rewards/Rewards.sol#3)
    - ABIEncoderV2 (contracts/rewards/Rewards.sol#4)
    - 0.6.11 (contracts/staking/Staking.sol#3)
    - ABIEncoderV2 (contracts/staking/Staking.sol#4)
    - 0.6.11 (contracts/token/PreToke.sol#3)
    - 0.6.11 (contracts/token/Toke.sol#3)
    - 0.6.11 (contracts/wallet/ZeroExTradeWallet.sol#3)
    - ABIEncoderV2 (contracts/wallet/ZeroExTradeWallet.sol#4)
```

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.

QSP-12 Sandwich Attacks Susceptibility in Sushiswap and Uniswap Controllers

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/controllers/SushiswapController.sol`, `contracts/controllers/UniswapController.sol`

Description: > A common attack in DeFi is the sandwich attack. Upon observing a trade of asset X for asset Y, an attacker frontruns the victim trade by also buying asset Y, lets the victim execute the trade, and then backruns (executes after) the victim by trading back the amount gained in the first trade. Intuitively, one uses the knowledge that someone's going to buy an asset, and that this trade will increase its price, to make a profit. The attacker's plan is to buy this asset cheap, let the victim buy at an increased price, and then sell the received amount again at a higher price afterwards.
In both stated contracts, `minAmountA` and `minAmountB` are provided in the `deploy` function. However, they are not checked after addLiquidity is called.

Recommendation: Sandwich attacks are hard to prevent reliably without a user-defined minOut or an oracle-computed price. It's recommended to keep the trade order size low relative to the pool's liquidity to make such attacks economically less attractive, or only use vaults for highly liquid tokens. Moreover, ensure that you add minAmount checks in both contracts.

QSP-13 Unsafe Cycle Durations Can be Set

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/manager/Manager.sol`

Description: The `cycleDuration`, which is set in the `initialize` and `setCycleDuration` functions, can be set to an unsafe small value. Since miners can manipulate block numbers within a small range, the miners might be able to trigger the `completeRollover` earlier or later than the intended `cycleDuration`.

Recommendation: Add a check in the setter function and make sure that the initialize function also uses the setter method to set the `cycleDuration`.

QSP-14 Some Functions do not Conform to Contract Pauses

Severity: Informational

Status: Fixed

File(s) affected: `contracts/pools/EthPool.sol`, `contracts/pools/Pool.sol`, `contracts/staking/Staking.sol`

Description: The `withdraw` function in the stated contracts, unlike the deposit function, is not pausable. The same holds true for the `slash` function in the staking contract.

Recommendation: Ensure that this is the intended behavior and add a comment in the code if this is the case. If this is a mistake, add a check in the corresponding function.

QSP-15 `schedule.setup` Checks Always Pass

Severity: Informational

Status: Acknowledged

File(s) affected: `contracts/staking/Staking.sol`

Description: Upon addition (`_addSchedule`) of schedules the setup property is set to true. The require statements in the `slash` and `depositFor` functions seem to be useless.

Recommendation: Consider removing the setup property, if it is not needed.

QSP-16 Schedules Cannot be Modified

Severity: Informational

Status: Fixed

File(s) affected: `contracts/staking/Staking.sol`

Description: There is no possiblity to alter properties of a given schedule. It might be needed to set certain props, namely `isActive` and `isPublic`. Not being able to unset `isActive` seems to be a bigger issue. To stop deposits, the only possibilities would be i) to pause the contract (which would affects all schedules) or ii) remove the schedule of interest.

Recommendation: Ensure that the missing logic is not needed.

QSP-17 ZeroExTradeWallet is Built Upon High Trust in Router

Severity: Informational

Status: Fixed

File(s) affected: `contracts/wallet/ZeroExTradeWallet.sol`

Description: All the deposited tokens are approved with the maximum value to be pulled out by the `zeroExRouter` address.

Recommendation: If possible, reconsider the design such that the contract routes signed requests to zeroEx directly. If this is a design decision, consider issuing approvals at the transaction level instead giving a max approval.

QSP-18 Sushiswap funds are not eligible for SUSHI reward

Severity: Informational

Status: Fixed

File(s) affected: `contracts/controllers/SushiswapController.sol`

Description: The provided liquidity is not eligible for SUSHI rewards.

Recommendation: To obtain rewards, you should deposit through the masterchef contract as well.

QSP-19 Privileged Roles and Ownership

Severity: Informational

Status: Acknowledged

File(s) affected: `ZeroExTradeWallet.sol`, `Toke.sol`, `Rewards.sol`, `EthPool.sol`, `Manager.sol`, `Staking.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. The owner can pause contracts and perform privileged operations. For instance, the owner can withdraw the diff of already withdrawn and initially vested tokens to the treasury. Users can lose funds if the owner account gets compromised or they go rogue.

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.

QSP-20 ZeroExTradeWallet Accepts External Token Addresses and Triggers Untrusted Code

Severity: Undetermined

Status: Fixed

File(s) affected: `contracts/wallet/ZeroExTradeWallet.sol`

Description: Both `deposit` and `withdraw` functions accept external addresses and trust them with calling transfer (safeTransfer) on them. Calling token methods on arbitrary addresses is a bad security practice. At the moment, only the manager can call these methods. Moreover, deposit does not check duplicates before adding tokens to the token list.

Recommendation: Consider whitelisting allowed token addresses. Prevent duplicates by adding a check.

Automated Analyses

Slither

We manually analysed the detected security issues. All the reported security issues are deemed as false positive.

Code Documentation

The contracts come with limited inlined comments. This would make the understanding of the code more difficult.

Adherence to Best Practices

1. EthPool and Pool are almost identical; EthPool support ERC20 transfers as well. It would make more sense to drop Pool.sol and use EthPool.sol for both the ERC20 and ETH pools.

2. in `contracts/pools/EthPool.sol` and `contracts/pools/Pool.sol` instead of overriding transfer and transferFrom to add a call to `preTransferAdjustWithheldLiquidity` function you should simply use the `_beforeTokenTransfer` hook

3. in `contracts/pools/EthPool.sol`, `contracts/pools/Pool.sol` and `contracts/staking/Staking.sol:179~183`: `solidity=121 if (manager.getRolloverStatus()) { requestedWithdrawals[msg.sender].minCycle = manager.getCurrentCycleIndex().add(2); } else { requestedWithdrawals[msg.sender].minCycle = manager.getCurrentCycleIndex().add(1); }` Consider adding comments corresponding to add(2) and add(1) and use constants instead of magic numbers.

4. TODO items in the code indicate an incomplete contract. Ensure that all the TODOs are done before the deployment:

   - `UniswapController.sol#60`
   - `UniswapController.sol#88`
   - `SushiswapController.sol#59`

- SushiswapController.sol#87
- EthPool.sol#33
- Pool.sol#63

5. Consider adding `nonreentrant` guards to all deposit and withdraw functions

Test Results

Test Suite Results

All tests pass.

```
Network Info
============
> HardhatEVM: v2.6.1
> network:    hardhat

No need to generate any newer typings.


  Test CoreEvent
Warning: Potentially unsafe deployment of Manager

    You are using the `unsafeAllow.delegatecall` flag.

    Full Runs
       ✓ Forces funds to be withdrawn and sends to farm based on rates (23806ms)
       ✓ An early transfer to treasury still allows appropriate withdraw by users (2131ms)

  Test Defi-Round
Warning: Potentially unsafe deployment of Manager

    You are using the `unsafeAllow.delegatecall` flag.

     ✓ Can submit ETH but get WETH during withdraw (12995ms)
    Deposit
       ✓ Requires pre-approval (1739ms)
       ✓ Converts ETH to WETH and deposits to contract (97ms)
       ✓ Updates total value (219ms)
    Stage 2
       ✓ WETH converted back to ETH on request (164ms)
       ✓ Can submit ETH but get WETH during withdraw (168ms)
       ✓ Can submit WETH and get WETH during withdraw (489ms)
       ✓ Can submit WETH and get ETH during withdraw (172ms)
    Finalize Assets
       ✓ Reverts when transfer to treasury is not complete (110ms)
       ✓ Reverts when no over-subscription (3200ms)
       ✓ Allows a rate such that all funds could be pulled to treasury in an emergency (189ms)
       ✓ Setting depositToGenesis to true (4947ms)
       ✓ Using just ETH when in range with round numbers (213ms)
       ✓ Using just USDC when in range with round numbers (957ms)
       ✓ Using just USDC when in range (202ms)
       ✓ Using just USDC when in range (221ms)
       ✓ Using just ETH when over (201ms)
       ✓ Using just USDC when over (211ms)
       ✓ Is prevented when transfer to treasury hasn't occured (151ms)
    Full Cycle
       ✓ Funds end up in treasury when in range (1033ms)
       ✓ Funds end up in treasury when oversubscribed (258ms)

  ETH Pool Test
Warning: Potentially unsafe deployment of Manager

    You are using the `unsafeAllow.delegatecall` flag.

    Test Initializer
       ✓ Test defaults
    Test Deposit
       ✓ Test deposit reverts on 0 amount
       ✓ Test deposit is successful (761ms)
       ✓ Deposits blocked when paused (51ms)
       ✓ Test eth deposit is successful
       ✓ Test eth deposit is fails if amount and value are mismatched
       ✓ Test multiple deposits from various addresses is successful (373ms)
    Test Deposit For
       ✓ Test deposit reverts on 0 amount
       ✓ Test deposit reverts on zero address
       ✓ Test deposit is successful
       ✓ Deposits blocked when paused (47ms)
       ✓ Test deposit is successful on behalf of another
       ✓ Test deposit is unsuccessful on behalf of another user without prior approval
       ✓ Test eth deposit is successful
       ✓ Test eth deposit is fails if amount and value are mismatched
       ✓ Test multiple deposits from various addresses is successful (62ms)
    Request Withdrawal
       ✓ reverts on 0 amount
       ✓ reverts on insufficient user balance
       ✓ withdraw request is successful (41ms)
       ✓ second withdraw request overwrites prior request (59ms)
       ✓ second withdraw request overwrites prior request after cycle moves forward (72ms)
    Withdrawal
       ✓ withdraw is unsuccessful when attempting to withdraw more than was requested (54ms)
       ✓ withdraw is unsuccessful when amount is 0 (52ms)
       ✓ withdraw is unsuccessful for an invalid cycle (47ms)
       ✓ a partial withdraw is successful when the pool has sufficient balance (76ms)
       ✓ a full withdraw is successful when the pool balance is sufficient (83ms)
      - a withdraw is allowed event when contract is paused
       ✓ a full withdraw in eth is successful when the pool balance is sufficient (88ms)
       ✓ Withdraw is reverted when pool balance is insufficient (849ms)
    Manage Approval
       ✓ Test approve manager not by owner
       ✓ Test approve manager by owner
       ✓ Can set allowance to 0
       ✓ Can set allowance to max (52ms)
       ✓ Can increase allowance (52ms)
       ✓ Can decrease allowance (38ms)
       ✓ Can set allowance to itself (53ms)
    Pool ERC20 transfers
       ✓ Decrements withdraw amount able to be requested (39ms)
       ✓ Removes withdraw request when amount has been wiped by transfer (54ms)
       ✓ Decrements withdraw requests already in progress (71ms)
       ✓ Decrements withdraw requests already in progress but allows remaining (97ms)
       ✓ Frees up total witheld liquidity (57ms)
    Pool ERC20 transferFroms
       ✓ Decrements withdraw amount able to be requested (49ms)
       ✓ Removes withdraw request when amount has been wiped by transfer (67ms)
       ✓ Decrements withdraw requests already in progress (79ms)
       ✓ Decrements withdraw requests already in progress but allows remaining (100ms)
       ✓ Frees up total witheld liquidity (63ms)

  Pool Test
    Test Deposit For
       ✓ Test deposit is successful on behalf of another (705ms)
       ✓ Test deposit is unsuccessful on behalf of another user without prior approval
    Manager Approval
       ✓ Can set allowance to 0 (356ms)
       ✓ Can set allowance to max (54ms)
       ✓ Can increase allowance (52ms)
       ✓ Can decrease allowance (40ms)
       ✓ Can set allowance to itself (52ms)

  Redeem
    Convert
       ✓ Moves funds to the correct schedule (69ms)

  Test CoreEvent
    Test Constructor
       ✓ Constructor adds correct supported tokens
    Test setDuration()
       ✓ Stores proper values
       ✓ Emits a DurationSet event
       ✓ Reverts when anyone but owner calls function
       ✓ Can only be called once
    Increase Duration
       ✓ Can only be called once started
       ✓ Cannot be used to decrease the duration
       ✓ Duration can be increased event after initial expiration
       ✓ Duration cannot be increased once a rate has been set (46ms)
       ✓ Duration cannot be increased once a token has been set to not swap
       ✓ Duration can be increased
       ✓ Increasing duration does not change starting block
       ✓ Emits event with updated information
    Test addSupportedTokens()
       ✓ Reverts when no tokens are submitted
       ✓ Reverts on duplicate
       ✓ Deployer can add supported token
       ✓ Emits a SupportedTokensAdded event
       ✓ Reverts when anyone but owner calls function
       ✓ Forces system finalized to be false
    Test deposit() revert when duration not set
       ✓ Reverts when event duration has not been set
    Test deposit()
       ✓ Reverts when duration has been reached
       ✓ Does not revert when within deposit period
       ✓ Reverts when TokenData array is empty
       ✓ Reverts on unsupported token
       ✓ Reverts on supported and unsupported tokens (40ms)
       ✓ Reverts on 0 amount
       ✓ Reverts on 0 amount with multiple tokens
       ✓ Reverts on overage
       ✓ Reverts with overage on one of two tokens
       ✓ Reverts when second deposit of singular asset exceeds limit
       ✓ Should emit a 'Deposited' event with the proper args (43ms)
       ✓ Should deposit the correct amount for a single deposit
       ✓ Should store balances of multiple tokens (53ms)
       ✓ Correct deposits for multiple deposits of same token (86ms)
    Test withdraw()
       ✓ Reverts when rates are locked
       ✓ Reverts on empty array
       ✓ Reverts on zero balance
       ✓ Reverts with both zero and non-zero balance (84ms)
       ✓ User cannot withdraw without deposit
       ✓ Reverts on zero address
       ✓ Does not allow for a withdrawal greater than a deposit
       ✓ Reverts onv multiple withdrawals, one overage (84ms)
       ✓ Should emit a 'Withdrawn' event (44ms)
       ✓ Stores the correct balances after a withdrawal (79ms)
    Set Rates
       ✓ Forces you to wait until the round ends (42ms)
       ✓ Allows an empty pool
       ✓ Doesn't allow any rates with a 0 in numerator or denominator (65ms)
       ✓ Can only set rates for supported tokens (47ms)
       ✓ Can delete a rate if its not finalized (169ms)
       ✓ Can update a rate if its not finalized (183ms)
    Test Finalize
       ✓ Reverts if deposit / withdraw period has not ended
       ✓ Reverts when TokenFaming array is empty
       ✓ Reverts when treasury transfer is not complete
       ✓ Reverts on zero address (62ms)
       ✓ Reverts on insufficient funds (102ms)
       ✓ Prevents a user from finalizing more than once (99ms)
       ✓ Reverts when ineffective amount is 0 (84ms)
       ✓ Reverts if pool address is zero address (86ms)
    Test transferToTreasury()
       ✓ Reverts when deposit / withdraw period has not ended yet
       ✓ Reverts when rates havent been published
       ✓ Emits a 'TreasuryTransfer' events with correct args (79ms)
       ✓ Wont allow a treasury transfer for the same token more than once (87ms)
    Get Account Data
       ✓ Token is populated regardless of deposit made
    Whitelist
       ✓ Allows included user to deposit
       ✓ Blocks someone from using anothers proof
       ✓ Disabled whitelist check lets anyone through
    Set No-Swap
       ✓ Is only callable after deposit/withdraw period
       ✓ Only allows supported tokens
       ✓ Is only callable once per token
       ✓ Emits event
    Rate and Treasury State Transition
       ✓ Won't allow finalization before end of duration
       ✓ Won't allow rate change once finalized via treasury transfer (107ms)
       ✓ Won't allow rate change once finalized via no-swap (88ms)
       ✓ Won't allow a no-swap set if a rate has already been published (127ms)

  Test Defi-Round
    Publish Rates
       ✓ Accepts and returns data when in range (136ms)
       ✓ Accepts and returns data when in range with 2 precision TOKE price (125ms)
       ✓ Accepts and returns data when in range with 8 precision TOKE price (132ms)
       ✓ Accepts and returns data when min subscribed (131ms)
       ✓ Accepts and returns data when max subscribed (128ms)
       ✓ Emits a RatesPublished event (64ms)
    Deposit
       ✓ With ETH only calculates correct account value (222ms)
       ✓ With DAI only calculates correct account value (232ms)
       ✓ With USDC only calculates correct account value (239ms)
       ✓ Should emit a Deposited event (272ms)
       ✓ Should revert when attempting to withdraw 0 tokens (227ms)
BigNumber { _hex: '0x00', _isBigNumber: true }
       ✓ Should allow user to withdraw deposited tokens (339ms)
       ✓ Should allow for a partial withdrawal
       ✓ Should allow for a withdrawal in Eth instead of WETH
    Withdraw
       ✓ Should revert when in stage 1 (122ms)
       ✓ Should revert when an unsupported token is claimed (205ms)
       ✓ Should revert when attempting to withdraw 0 tokens (215ms)
       ✓ should revert when a user withdraws more than their balance (304ms)
       ✓ Should allow user to withdraw deposited tokens (338ms)
       ✓ Should allow for a partial withdrawal (329ms)
       ✓ Should not allow you to deposit one token and withdraw another (313ms)
    Get Total Value
       ✓ Calculates when only single coin deposited (279ms)
    Going to Stage 2
       ✓ fails when called by not owner
       ✓ fails if last look duration has not passed
       ✓ fails if overNumerator is not greater than 0
       ✓ fails if overdenominator is not greater than 0
       ✓ passes if all conditions are satisfied
       ✓ emits a RatesPublished event
    Get Account Data
       ✓ correctly fetches account data if there was never any deposit
       ✓ correctly fetches account data amongst all deposits (211ms)
       ✓ Returns even when rates not published (372ms)
       ✓ Returns effective/ineffective amounts after rates are published (342ms)
    Whitelist
       ✓ Allows included user to deposit (186ms)
       ✓ Blocks someone from using anothers proof (152ms)
       ✓ Disabled whitelist check lets anyone through (188ms)
    Adding Supported Tokens
       ✓ Should not allow another user to add a supported token
       ✓ Should allow deployer to add supported tokens
       ✓ Emits a SupportedTokensAdded event
    Getting supported tokens
       ✓ Should return the correct token addresses
    Getting Genesis Pools
       ✓ Should return the correct genesis addresses
    Getting oracle addresses
       ✓ Should return the correct oracle addresses

  Test Manager
Warning: Potentially unsafe deployment of Manager

    You are using the `unsafeAllow.delegatecall` flag.

    Test Initializer
       ✓ Test defaults
    Register Controller
       ✓ register controller fails when not by admin
       ✓ register controller successfully
    Unregister Controller
       ✓ unregister controller fails when not by admin
       ✓ unregister controller successfully
       ✓ updates controller registery (44ms)
    Register Pool
       ✓ regiser pool fails when not by admin
       ✓ register a duplicate pool fails
       ✓ register pool successfully
    Unregister Pool
       ✓ unregiser pool fails when not by admin
       ✓ unregister a pool that does not exist fails
       ✓ unregister pool successfully
    Cycle Duration
       ✓ set cycle duration fails when not by admin
       ✓ set cycle duration successfully
    Cycle Completion
       ✓ complete rollover fails when not by rollover role
       ✓ prevents premature execution
       ✓ allows properly timed execution
       ✓ sets current cycle to current block
       ✓ increment currentCycleIndex
       ✓ to emit cycle rollover complete event
  Starting Cycle Rollover
     ✓ set the rolloverStarted flag to true
  Execute Maintenance
     ✓ Call fails when user doesn't have proper role
     ✓ Call succeeds when user has proper role

Test Pool
  Test Initializer
     ✓ Test defaults
  Test Deposit
     ✓ Test deposit reverts on 0 amount
     ✓ Test deposit is successful
     ✓ Deposits blocked when paused (52ms)
     ✓ Test multiple deposits from various addresses is successful (60ms)
  Test Deposit For
     ✓ Test deposit reverts on 0 amount
     ✓ Test deposit reverts on zero address
     ✓ Test deposit is successful
     ✓ Deposits blocked when paused (52ms)
     ✓ Test multiple deposits from various addresses is successful (61ms)
  Request Withdrawal
     ✓ reverts on 0 amount (48ms)
     ✓ reverts on insufficient user balance (50ms)
     ✓ withdraw request is successful (91ms)
     ✓ second withdraw request overwrites prior request (107ms)
     ✓ second withdraw request overwrites prior request after cycle moves forward (114ms)
  Withdrawal
     ✓ withdraw is unsuccessful when attempting to withdraw more than was requested (140ms)
     ✓ withdraw is unsuccessful when amount is 0 (124ms)
     ✓ withdraw is unsuccessful for an invalid cycle (117ms)
     ✓ a partial withdraw is successful when the pool has sufficient balance (147ms)
     ✓ a partial withdraw is unsuccessful when the pool has insufficient balance for the requested amount (129ms)
     ✓ a full withdraw is successful when the pool balance is sufficient (147ms)
    - a withdraw is allowed even when contract paused
  Test manager approval
     ✓ Test approve manager not by owner
     ✓ Test approve manager by owner (49ms)
  Pool ERC20 transfers
     ✓ Decrements withdraw amount able to be requested (49ms)
     ✓ Removes withdraw request when amount has been wiped by transfer (113ms)
     ✓ Decrements withdraw requests already in progress (105ms)
     ✓ Decrements withdraw requests already in progress but allows remaining (484ms)
     ✓ Frees up total witheld liquidity (118ms)
  Pool ERC20 transferFroms
     ✓ Decrements withdraw amount able to be requested (58ms)
     ✓ Removes withdraw request when amount has been wiped by transfer (119ms)
     ✓ Decrements withdraw requests already in progress (112ms)
     ✓ Decrements withdraw requests already in progress but allows remaining (163ms)
     ✓ Frees up total witheld liquidity (125ms)

Test PTOKE Token
  Initialization
     ✓ Has a name
     ✓ Has a symbol
     ✓ Has standard precision
     ✓ Mints 10 to User 1
     ✓ Deployer has ADMIN, MINTER, and PAUSER Roles
     ✓ Nobody else has ADMIN, MINTER, and PAUSER Roles

Contract: Rewards Unit Test
  Test Constructor
     ✓ Test Invalid TOKE Address (749ms)
     ✓ Test Invalid Signer Address
  Test Defaults
     ✓ Test TOKE Contract set
     ✓ Test Signer set
     ✓ Test Owner is set
  Test Setters
     ✓ Test setting signer by not owner
     ✓ Test setting signer to 0 address fails
     ✓ Test setting signer
  Test Claiming
     ✓ Test Signature mismatch (478ms)
     ✓ Test claimable must be greater than 0
     ✓ Test claiming more than available balance of contract (66ms)
     ✓ Test claiming for another user (42ms)
     ✓ Test all funds can be removed from contract (51ms)
     ✓ Test claiming when signer changes (92ms)
     ✓ Test Claim event is emitted (50ms)
     ✓ Test reuse of signature (68ms)
     ✓ Test successful signature and valid transfer (53ms)
     ✓ Test amounst are accumulative (85ms)
     ✓ Claimed balances are tracked (60ms)

Test Staking
  Deposit
     ✓ Test deposit reverts on 0 amount
     ✓ Updates initial amounts on staking schedules (60ms)
     ✓ Amounts deposited to 0 schedule are immediately staked (55ms)
     ✓ Amounts deposited to 0 schedule are immediately available for withdrawal (56ms)
     ✓ Are blocked when contract is paused (77ms)
  Withdrawal
     ✓ withdraw is unsuccessful when attempting to withdraw more than was requested (131ms)
     ✓ withdraw is unsuccessful when amount is 0 (130ms)
     ✓ withdraw is unsuccessful for an invalid cycle (130ms)
     ✓ a partial withdraw is successful when the stakingContract has sufficient balance (146ms)
     ✓ a full withdraw is successful when the stakingContract balance is sufficient (161ms)
     ✓ Does not allow a withdrawal when paused (134ms)
  Request Withdrawal
     ✓ reverts on 0 amount (50ms)
     ✓ reverts on insufficient user balance (62ms)
     ✓ withdraw request is successful (96ms)
     ✓ second withdraw request overwrites prior request (113ms)
     ✓ second withdraw request overwrites prior request after cycle moves forward (133ms)
  Adding Schedules
     ✓ Increments index (38ms)
     ✓ Forces setup to always be true
     ✓ Enforces minimum duration
     ✓ Enforces minimum interval
  Removing Schedules
     ✓ Forces schedule to exist
     ✓ Manages internal structures (67ms)
     ✓ Schedule can't be removed twice
  Long Term Staking
     ✓ Has 0 staked if before the cliff (77ms)
     ✓ Staks completely after the cliff and duration (74ms)
     ✓ Stakes according to interval (97ms)
  Long Term Staking and Withdrawals
     ✓ Blocks request if 0 staked (103ms)
     ✓ Allows request of partially staked amounts (112ms)
     ✓ Allows request of partially staked amounts across multiple schedules (140ms)
     ✓ Withdraw of partially staked amounts across multiple schedules is tracked (218ms)
     ✓ Deleting a schedule allows immediate withdraw of funds (209ms)
  Slashing
     ✓ Enforces valid amount
     ✓ Enforces valid schedule
     ✓ Can only slash when a deposit was made first
     ✓ Checks the stak has sufficient initial balance (65ms)
     ✓ Withdrawn amounts cannot be slashed (160ms)
     ✓ Slashed tracking is additive (116ms)
     ✓ Withdraw requests are slashable before they're executed (199ms)
     ✓ Slashed amounts cannot be withdrawn (176ms)
     ✓ Slashing decreases your balance (79ms)
     ✓ Slashing does not decrease your staking amounts (107ms)
     ✓ Slashing your entire amount before it vests won't underflow availableForWithdrawal (84ms)
     ✓ Slashing is not allowed when the contract is paused (74ms)
  Withdraw Requests During Rollover
     ✓ Is allowed but has to wait 2x cycles (164ms)
  Permissioned Schedules
     ✓ Blocks not permitted callers (112ms)
     ✓ Permissioned explicit can add on the fly (90ms)
     ✓ Owner can add schedule on the fly (51ms)
     ✓ Depositors permissions can be removed
  Test setScehduleStatus()
     ✓ Should set isActive boolean

Test TOKE Token
  Initialization
     ✓ Has a name
     ✓ Has a symbol
     ✓ Has standard precision
     ✓ Mints 100M
     ✓ Mints initial amount to deployer
  Pausing
     ✓ Prevents transfers when paused (38ms)

Test ZeroExTradeWallet
  Constructor Arguments
     ✓ constructor fails on router 0 address (330ms)
     ✓ constructor fails on manager 0 address
  Initialization
     ✓ router is set properly
     ✓ manager is set properly
     ✓ owner is the deployer
  Whitelisting Tokens
     ✓ Does not let anyone but the owner whitelist tokens
```

```
        ✓ Returns proper whitelisted tokens
        ✓ Does not add duplicate addresses
     Removing Whitelisted Tokens
        ✓ Should remove one of two whitelisted tokens
     Deposit
        ✓ deposit fails when not by manager
        ✓ initial deposit succeeds by manager (174ms)
        ✓ subsequent deposit succeeds by manager (185ms)
        ✓ Reverts when non-whitelisted token address passed in (58ms)
     Withdraw
        ✓ withraw fails when not by manager
        ✓ withdraw succeeds by manager (240ms)
       - withdraw succeeds by manager and zero's out token list
        ✓ Reverts when non-whitelisted token address passed in (58ms)
     Signer Authorization
        ✓ fails when registering signer as 0 address
        ✓ Can successfully call the router (1764ms)

   Test Manager
 Warning: Potentially unsafe deployment of Manager

     You are using the `unsafeAllow.delegatecall` flag.

     Operation and Timing Checks
        ✓ Only allows liquidity pulls from pool during rollover period (1586ms)
        ✓ Only allows liquidity pushes to pool during rollover period (76ms)
        ✓ Allows controller operations mid-cycle (21966ms)
     Full Execution
        ✓ Runs (3741ms)


   349 passing (3m)
   3 pending
```

Code Coverage

The reported branch coverage is 63.27%. We recommend to aim for a 100% branch coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
Imports.sol	100	100	100	100	
contracts/airdrop/	0	0	0	0	
AirdropPush.sol	0	0	0	0	18,19,20
contracts/controllers/	55.07	34.62	64.71	55.07	
BalancerController.sol	92.31	50	100	92.31	60
SushiswapController.sol	0	0	0	0	… 121,122,124
UniswapController.sol	88.24	50	100	88.24	100,103
ZeroExController.sol	84.62	50	100	84.62	53,56
contracts/core/	99.28	57.95	100	98.59	
CoreEvent.sol	99.28	57.95	100	98.59	266,340
contracts/defi-round/	97.4	58.89	90.91	96.13	
DefiRound.sol	97.4	58.89	90.91	96.13	… 332,336,365
contracts/interfaces/	100	100	100	100	
ICoreEvent.sol	100	100	100	100	
IDefiRound.sol	100	100	100	100	
ILiquidityEthPool.sol	100	100	100	100	
ILiquidityPool.sol	100	100	100	100	
IManager.sol	100	100	100	100	
IStaking.sol	100	100	100	100	
IWETH.sol	100	100	100	100	
IWallet.sol	100	100	100	100	
contracts/interfaces/balancer/	100	100	100	100	
IBalancerPool.sol	100	100	100	100	
IBalancerRegistry.sol	100	100	100	100	
contracts/manager/	95.95	60	95.45	94.94	
Manager.sol	95.95	60	95.45	94.94	137,173,204,216
contracts/pools/	97.25	77.14	100	97.39	
EthPool.sol	98.28	89.47	100	98.36	127
Pool.sol	96.08	62.5	100	96.3	115,192
contracts/redeem/	75	40	66.67	75	
Redeem.sol	75	40	66.67	75	56,62,63,64
contracts/rewards/	95.65	50	100	95.83	
Rewards.sol	95.65	50	100	95.83	135
contracts/staking/	99.29	85.94	96.3	99.3	
Staking.sol	99.29	85.94	96.3	99.3	103
contracts/token/	66.67	100	100	66.67	
PreToke.sol	100	100	100	100	
Toke.sol	66.67	100	100	66.67	20
contracts/wallet/	97.22	57.69	100	100	
ZeroExTradeWallet.sol	97.22	57.69	100	100	
All files	**93.08**	**63.27**	**92.41**	**92.87**	

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

```
4616ee795324c414716adcb5e107fff694584e7a4f1e023137cbf28d307e596d ./contracts/interfaces/IWallet.sol
df0ddace1bd7e37c8639deeaffe4b9c2e13a6c706fd2aa2085a90863f8a45faa ./contracts/interfaces/ILiquidityEthPool.sol
d357c8bba442759f0e427fe6b2386cc592bd1de8e56e129445b137d2271a5f70 ./contracts/interfaces/IWETH.sol
a1da805838da13f6c9e8fec27c795e223d215cf88736d8b2cde002add9748853 ./contracts/interfaces/ILiquidityPool.sol
feb86b4db006a833181a81cdbfd4f8d5edfe91986d9cf47aecd1a62de506b06d ./contracts/interfaces/IStaking.sol
58639ac77c6347a6d2b2db239be5b16b3c5f1d5c039037e250bf07243cb2d02f ./contracts/interfaces/IManager.sol
963c4f11934577866c9c7b9d0905e29e8b91e56e7a1885cc8ac3f17a023f37bc ./contracts/interfaces/balancer/IBalancerRegistry.sol
70659d16e77058f914bd730ca4213cce4d34315d1471c466a32e375e12863208 ./contracts/controllers/BalancerController.sol
3462b46454d95e2ea8ea707d6fdebbbd4c24636f7274e88ec44fe482e2a60601 ./contracts/controllers/ZeroExController.sol
c4fa047c9aa6b4fbfe2f51e5846417f5d50355c464c56eb2d09e4ce6ed9bbd67 ./contracts/controllers/UniswapController.sol
ece8a0974c7288dc5853cf21498770d779de9d14516b17fea39f90ba35c617c8 ./contracts/controllers/SushiswapController.sol
16a22f11ce8cd5b86d615d1ea0b73afaf614d13b380665e5938cdab6c4b7840a ./contracts/wallet/ZeroExTradeWallet.sol
d43a5384ae10d5e4fe8506d50927d980da38e066b06798262f0e36b1ec3c5085 ./contracts/staking/Staking.sol
a23436b8c9b6c66b3cdf60e530e04958961ebfcd6e3dd01e5c8d12d740375380 ./contracts/manager/Manager.sol
e7eda61e9f96c98ea07a77b1a5d03c844803095d57ee1e7e1cc4bad7933be687 ./contracts/rewards/Rewards.sol
476fa773e90a936872dd69932b977743431db3fb59eb4a04bdbc9ba69713ea8d ./contracts/pools/Pool.sol
fb175162636659864ea9b28201abad33b67a8a157db147fc67a4e27a810edc09 ./contracts/pools/EthPool.sol
ff3178765b785bdc5e79f6a5393e1c91bd2c65fc3294c3678e75d9df9ec269ff ./contracts/token/Toke.sol
```

Tests

```
8c67742717df1380434dcd98421dbe2836685d21ee78e84794b1b01da74027fc ./ProxyAdmin.test.js
d1911a62f09847fbb6409ee9cb465178ad83f306118fefc829927c6051046420 ./ZeroExTradeWallet.test.js
f8134206dc4b064a04070b56e82c52c95f5b8bc9ec317f5c2e0e65d8d280f6d9 ./Staking.test.js
5baa0c751b41e5428bd514f6bdba85f02e7ed9d4fcd928a44c734c728822e7fe ./Pool.test.js
7be810dbdf0d4b6304da3ab0b6cef0e3dc41402fde810b85da6252078e51956b ./Defi-Round.test.js
6bbe2cbed9a050999c4cfc467318c1fd74831ddb7791ea2fb16be616b4a1b12c ./Toke.test.js
643815e773d89494f8c566dac73c44944c812407b8eccf3034d5e13305a2c4b4 ./Rewards.test.js
bc5aeb0f2cebf02136d103f64016ec7626c29c84b171372ba987aa5ae61796f8 ./PreToke.test.ts
112c406d967bd0dfe152cfbc987690590b42323baa1bbcc81dea572ad7f3824e ./Manager.test.js
9c18e6a89c95cecd17b3edb1a47d42b8ff1035ea1567d36b26315a5b0d34aae5 ./CoreEvent.test.ts
3bfa8b6c7dc4147b9fa9c2211ba99ace2750e154cd861f176142df8bc0953be3 ./test-integration/ETHPool.test.ts
31455cffb36954bb2c266b2bddabae46f7bc30680fe7f20af51c14bd6e3d673b ./test-integration/CoreEvent.test.ts
689320b14740a15d005ecf3c46cd39eb7c0bfa28222cf0353dc3e38547c658be ./test-integration/Pool.test.js
5a249041fa8bc77398a6e4d9f4a06d53d9086b240fb59964994040e689e4ad00 ./test-integration/Redeem.test.ts
7afb776e10dd2412bfa0020ccc2a8fa2c68a8569bdb05d6ec8b2c90e654d3407 ./test-integration/Defi-Round.test.js
a045e65d9c79082a5bc3e027662770f4b363de5a3859769f80621213565e7d07 ./test-integration/manager/Manager.test.js
a8a7e80d18d2794d33288073768b0be49a128d34ea8137ce8ceee75ea9d7f477 ./test-integration/manager/deployWithdraw/uniV2Base.js
0948ac261704853b1593cda84a58487964334b6832e1276c9f729fcf61031b09 ./test-integration/manager/deployWithdraw/balancerTest.js
7dd07ca8abc16ab6c031dfb1f39208c3ec843f51e1d8aab0df3c5e2cc35e4c6a ./test-integration/manager/deployWithdraw/sushiV2Test.js
186321a9856995dfee42c630587146b06be5d2be395c636cedddf2fa4b3b5a71 ./test-integration/manager/deployWithdraw/uniV2Test.js
d113d2a14b009e000e896a68cddb6c34472cd697169879fdbb733d52f85aefb0 ./test-integration/manager/deployWithdraw/zeroExTest.js
```

Changelog

- 2021-07-29 - Initial report
- 2021-08-31 - Reaudit

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.

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.