Quantstamp Security Assessment Certificate

June 24th 2022 — Quantstamp Verified

Klap - Lockdrop

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

Executive Summary

Type	Defi
Auditors	Andy Lin, Senior Auditing Engineer Ed Zulkoski, Senior Security Engineer Bohan Zhang, Auditing Engineer
Timeline	2022-06-24 through 2022-06-24
EVM	Arrow Glacier
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Low
Test Quality	Low
Source Code	

Repository	Commit
lending-contracts	6be12136

Total Issues	**8** (0 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	**5** (0 Resolved)
Informational Risk Issues	**2** (0 Resolved)
Undetermined Risk Issues	**1** (0 Resolved)

8 Unresolved
0 Acknowledged
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).
○ Fixed	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

Lockdrop is a contract that collects the users' deposit up to a USD cap and allows the users to claim those AToken back after a certain period. The code is simple and of good quality. This audit's main concerns are privileged roles and token price updates.

ID	Description	Severity	Status
QSP-1	Price Update Does Not Account for Previously Deposited Tokens	⌄ Low	Unresolved
QSP-2	Privileged Roles and Ownership	⌄ Low	Unresolved
QSP-3	Variable Inconsistency	⌄ Low	Unresolved
QSP-4	Unchecked Return Value	⌄ Low	Unresolved
QSP-5	Locked Atoken Interest	⌄ Low	Unresolved
QSP-6	Block Timestamp Manipulation	○ Informational	Unresolved
QSP-7	Infinite Deposit Period	○ Informational	Unresolved
QSP-8	Price Update Scheme Not Robust to Price Fluctuations	? Undetermined	Unresolved

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

DISCLAIMER:
The scope of this audit is only the file `staking/Lockdrop.sol`.

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 v0.8.3

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither ./contracts/staking/Lockdrop.sol --solc-remaps @openzeppelin=node_modules/@openzeppelin`

Findings

QSP-1 Price Update Does Not Account for Previously Deposited Tokens

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: If `aTokens` have already been deposited into the contract and the price is updated, the `usdDeposited` amount is not adjusted to reflect the price change. For example, if the price of a deposited `aToken` changes from `$200` to `$100`, one would expect the `usdDeposited` amount to be reduced by `$100` for every deposited token. Currently, the `usdDeposited` amount is unaffected.

Recommendation: Either acknowledge that such fluctuations are acceptable for the `Lockdrop`, or consider adding a `mapping` that tracks the total deposits for each `aToken`. If a price update occurs, the `usdDeposited` amount would change according to the delta in price. For example, if 5 `aToken` valued at `$200` each have been deposited and the price is updated to `$100`, then the `usdDeposited` would be reduced by `5 * ($200 - $100) = $500`.

QSP-2 Privileged Roles and Ownership

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: The contract has 4 setter functions only callable by the owner:`setUnlockTime`, `setStatus`, `setPrice`, and `setCap`. Importantly, the owner can invoke `setUnlockTime()` to move the `unlockTime` arbitrarily far into the future. This may effectively lock all tokens indefinitely.

Exploit Scenario: The following scenarios could happen when there is a malicious owner.

1. The owner can set `unlockTime = 0`, then any user can claim the balance. The owner can set `unlockTime = 2^256-1`, then any user cannot claim the balance.
2. The owner can set `depositReady = false` or `cap = 1` to forbid users to call `deposit()`

Recommendation: Limit centralization where possible, specifically removing the ability for funds to be locked indefinitely. A few potential ways to do this are to either: 1) remove the function entirely and set the `unlockTime` in the `constructor()`; 2) disallow calls to `setUnlockTime()` if `depositReady` has **ever** been set to `true`. The second case would ensure that any user that deposits funds will have a guaranteed unlock time.

QSP-3 Variable Inconsistency

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: For the current implementation, a token address may have a non-zero price in the variable `prices` but not exist in `aTokens` when `setPrice` is called unexpectedly.

Exploit Scenario: Suppose address `addr` is not in `aTokens`, which is `aTokens[addr] = false`. The owner calls `setPrice(addr, 1)` to make `prices[addr] = 1`. Thus, `prices` and `aTokens` are inconsistent with each other.

Recommendation: Either update `aTokens` and `decimals` together or revert when the `_aToken` is never set.

QSP-4 Unchecked Return Value

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: According to the ERC20 standard, the `transfer` and `transferFrom` functions return `true` if the transfer was successful and `false` otherwise. These functions do not need to revert if the transfer is not successful. However, there are two places in the contract where these return values are ignored:

1. `claim()` ignores return value by `transfer(msg.sender, accountBalance)` on L85
2. `deposit()` ignores return value by `transferFrom(msg.sender, address(this), _amount)` on L73

If the token contract signals an error by returning `false` instead of reverting, the functions above do not detect any error and proceed with their execution.

Recommendation: Use OpenZeppelin's `SafeERC20` library for functions such as `safeTransfer()`.

QSP-5 Locked Atoken Interest

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: AToken wallet balance will automatically increase over time as it will auto accumulate the interest (doc). However, the `claim` will only withdraw the original amount during the `deposit`. The contract will lock the extra balance from the AToken, and there is no way to retrieve out.

Recommendation: Please clarify if it is intended to lock the funds. Otherwise, please add the feature to retrieve the extra balances.

QSP-6 Block Timestamp Manipulation

Severity: Informational

Status: Unresolved

Related Issue(s): SWC-116

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes. If a smart contract relies on a timestamp, it must take this into account (In function `claim()`).

Recommendation: The team should clarify to users that 15 seconds difference would not affect the functionality and security of the protocol.

QSP-7 Infinite Deposit Period

Severity: Informational

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: The contract allows deposit even when the `unlockTime` is over.

Recommendation: Add a time limit to the deposit.

QSP-8 Price Update Scheme Not Robust to Price Fluctuations

Severity: Undetermined

Status: Unresolved

File(s) affected: `stacking/Lockdrop.sol`

Description: Currently, token prices are manually adjusted in the contract based on owner calls to `setPrice()`. If the actual price has changed significantly before the owner updates the value, the total amount of locked tokens may exceed/be lower than the `cap` value.

Exploit Scenario: When the `Lockdrop.sol` contract is deployed, the `aMoon` token is $1, and the cap is set to $100, expecting to be able to lock 100 `aMoon` tokens. After 10 days, the token pumps, and the price becomes $1000. Now, Alice can no longer deposit the `aMoon` token.

Recommendation: Either acknowledge that such fluctuations are acceptable for the `Lockdrop` or if a more robust approach is needed, consider a Chainlink oracle to query prices. Meanwhile, we recommend limiting the deposit to a short period to mitigate the risk of price fluctuation.

Automated Analyses

Slither

It founds 36 results, and most are false positives or out of the scope. We have added the valid ones into the report.

Code Documentation

- Add a document stating that the cap can be 0 in L21-22. If it is zero, it means uncapped.

Adherence to Best Practices

- Suggest adding events for `setPrice()`, `setStatus()`, `setUnlockTime()`, and `setCap()`. This will help those who are listening to this contract make better judgement.
- Remove the public visibility from the constructor. After solidity 0.7, the visibility of the constructor is ignored.
- In the `constructor()`, the code can be optimized to save gas.

```
for(uint256 i = 0; i < _aTokens.length; i++) {
    aTokens[_aTokens[i]] = true;
    prices[_aTokens[i]] = _prices[i];
    decimals[_aTokens[i]] = _decimals[i];
}
```

Could be changed to

```
for(uint256 i = 0; i < _aTokens.length; i++) {
    address token = _aTokens[i];
    aTokens[token] = true;
    prices[token] = _prices[i];
    decimals[token] = _decimals[i];
}
```

Changelog

- 2022-06-24 - Initial report

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.