SHERLOCK

SHERLOCK SECURITY REVIEW FOR

LiquiFi

Prepared For:	LiquiFi
Prepared By:	Sherlock
Lead Security Experts:	Christoph Michel, shw
Dates Audited:	May 15th - May 27th, 2022
Report Delivered:	June 6th, 2022

Introduction

LiquiFi is the "Carta for crypto & web3". They help teams automate their vesting schedules, payroll schedules, and cap table management.

This report focused on the V1 vesting contracts. There are 3 types of vesting contract implementations: TokenVesting.sol, StaggeredTokenVesting.sol, and WeightedTokenVesting.sol.

Each implementation also has their own respective factory contracts as well. These factory contracts are used by the frontend Dapp to clone a new vesting contract per company.

Additionally, each vesting implementation calls a module, which is used to calculate the number of tokens that are claimable by a beneficiary.

Scope

Branch: main (https://github.com/LiquiFi/vesting-contracts-v1)
Commit: d32accb5ebe09f2b3908067f69c7c6f690976011 (https://github.com/LiquiFi/vesting-contracts-v1/commit/d32accb5ebe09f2b3908067f69c7c6f690976011)
Contracts:

- lib/helper/Token.sol
- lib/VestPlans.sol
- modules/StaggeredVestingModule.sol
- modules/VestingModule.sol
- modules/VestingModuleRegistry.sol
- modules/WeightedVestingModule.sol
- payments/BatchPayments.sol
- vesting/BaseTokenVesting.sol
- vesting/TokenVesting.sol
- vesting/TokenVestingFactory.sol
- vesting/staggered/StaggeredTokenVesting.sol
- vesting/staggered/StaggeredTokenVestingFactory.sol
- vesting/weighted/WeightedTokenVesting.sol
- vesting/weighted/WeightedTokenVestingFactory.sol

Protocol Attributes

Test coverage: Very good
Quality of tests: Very high

Language: Solidity
Upgradeable: Yes
Assembly usage: No
Solc version: 0.8.11

Blockchain: Ethereum
L2s: Polygon (planned), Arbitrum (planned), Optimism (planned)

Findings

Each issue has an assigned severity:

- Informational issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgement as to whether to address such issues.
- Low issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Total Issues

Informational	Low	Medium	High
4	8	2	2

Issue H-01

Staggered vesting's getClaimableAmount computes wrong awardAmount

Severity

High

Vulnerability Detail

The getClaimableAmount function computes the awardAmount from the staggered vesting plan as _staggeredPlan.cliffAmount + (_staggeredPlan.cadence * _staggeredPlan.tokenAmountPerPeriod). This is incorrect because the cadence reflects the length (in seconds) of a period, whereas the token amount per period should be multiplied by the total number of periods.

Impact

The impact is that this value will be highly overestimated.

The current StaggeredVestingModule does indeed ignore this value for its computations and is not needed.

However, if a team implements their own module, relying on the correctness of the awardAmount in the module can be fatal and lead to vesting funds being stolen.

Code Snippet

StaggeredTokenVesting.sol#L144, StaggeredVestingModule.sol#L29

Tool used

Manual Review

Recommendation

It is recommended to change the design of the deposit and withdrawal process so that funds can be processed without a queue. Users should be able to process their deposits or withdrawals regardless of a specific order.

```
uint256 awardAmount = _staggeredPlan.cliffAmount +
(_staggeredPlan.totalVestingPeriods *
_staggeredPlan.tokenAmountPerPeriod);
```

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed

Issue H-02

Staggered vesting's _awardAmount is never verified when claiming tokens

Severity

High

Vulnerability Detail

The _awardAmount parameter can be chosen by the claimTokens caller. This value is never verified although it should equal _staggeredPlan.cliffAmount + (_staggeredPlan.totalVestingPeriods * _staggeredPlan.tokenAmountPerPeriod).
This unchecked value is then forwarded to the IModule(module).claimable call.

Impact

The current StaggeredVestingModule does indeed ignore this value for its computations and is not needed.
However, if a team implements their own module, relying on the correctness of the awardAmount in the module can be fatal and lead to vesting funds being stolen.

Code Snippet

StaggeredTokenVesting.sol#L170, StaggeredVestingModule.sol#L29

Tool used

Manual review

Recommendation

The claimTokens function does not need the _awardAmount as a parameter, it can be computed from the _staggeredPlan parameter, similar to getClaimableAmount.

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed

Issue M-01

getClaimableAmount calls return positive amounts even if user does not have vestings

Severity

Medium

Vulnerability Detail

The getClaimableAmount function of all vesting contracts should return the claimable amounts of the _beneficiary parameter. They compute a hash based on the beneficiary and other parameters but it’s not checked that this hash is actually in the merkle tree. Therefore, this function also returns positive claimable amounts for users that don’t have anything to claim.

It could be that third-party smart contracts rely on this function to correctly return the claimable amount that could be received through a claimTokens call, given the same parameters plus a valid merkle proof.

Impact

The function also returns positive claimable amounts for users that don’t have anything to claim.

Code Snippet

TokenVesting.sol#L118, WeightedTokenVesting.sol#L111, StaggeredTokenVesting.sol#L126

Tool used

Manual review

Recommendation

Third parties should not merely rely on the return value from getClaimableAmount but also verify whether the queried user or award hash exists.

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed

Issue M-02

Pausing functionality does not work

Severity

Medium

Vulnerability Detail

The vesting contracts inherit from PausableUpgradeable and the whenNotPaused modifier is used but there's no implementation to actually pause the contract.

Impact

If the contract owner wants to pause the contracts, they will be unable to.

Code Snippet

All vesting contracts

Tool used

Manual review

Recommendation

If pausing functionality is required, add public pause and unpause functions that call PausableUpgradeable's internal _pause/_unpause functions.

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed

Issue L-01

Unable to update moduleData in vesting contracts

Severity

Low

Vulnerability Detail

The owner can update the module in a vesting contract, but the moduleData passed in the module is not updatable. The owner might want to update the module data if they change the module to a custom one that needs some specific data input.

Code Snippet

BaseTokenVesting.sol#L58-L62

Tool used

Manual review

Recommendation

Suggest allowing the owner to update moduleData (in the BaseTokenVesting contract) if reasonable.

Protocol Team Comment

Acknowledged

Issue L-02

Logic error in getRevokedTimestamp when plan is archived

Severity

Low

Vulnerability Detail

If the plan is archived, getRevokedTimestamp returns archivedTimestamp no matter if the user's revoke timestamp is set or not. Even if a user's revoke timestamp is set before the archivedTimestamp, getRevokedTimestamp returns the plan's archive timestamp.

Impact

Third parties relying on the return value of getRevokedTimestamp may think the user is revoked in the further future, but actually, the user's effective revoke timestamp is closer.

Code Snippet

TokenVesting.sol#L79, WeightedTokenVesting.sol#L70, StaggeredTokenVesting.sol#L69

Tool used

Manual review

Recommendation

Suggest returning the minimum value of the user's revoke timestamp and the archive timestamp, as done in the _getClaimableAmount function.

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed

Issue L-03

Owner can always archive specific users before archiving the entire plan

Severity

Low

Vulnerability Detail

Although the archiveBeneficiaryPlan prevents the owner from revoking a user if the archivedTimestamp is set, the owner can always choose to revoke the user before archiving the plan.

Impact

As a result, some users can have an effective revoke timestamp before the plan's archive timestamp.

Code Snippet

TokenVesting.sol#L214, WeightedTokenVesting.sol#L205, StaggeredTokenVesting.sol#L265

Tool used

Manual review

Recommendation

Suggest documenting this possible scenario if this is allowed or intended.

Protocol Team Comment

Acknowledged

Issue L-04

Owner can lock tokens in the vesting contracts by pausing or archiving the plan

Severity

Low

Vulnerability Detail

Even if the plan is frozen, the owner is allowed to set the plan's archive timestamp (if not set before). If the vesting plan hasn't started yet, the owner could avoid paying users by setting the archive timestamp before the start time. If the vesting plan is ongoing, the owner can set it to the current timestamp to avoid paying the rest.

Impact

Either way would cause the funds to be locked in the vesting contract. Besides, the owner can pause on the contracts to achieve the same effect (the contract needs to implement public pause methods first).

Code Snippet

BaseTokenVesting.sol#L73-L82

Tool used

Manual review

Recommendation

Suggest documenting this possible behavior if it is allowed. Otherwise, avoid the owner calling archivePlan and pause if the plan is frozen. An alternative way is to renounce the ownership when freezing the contract.

Protocol Team Comment

Acknowledged

Issue L-05

Users cannot ensure there are sufficient funds in the vesting contracts

Severity

Low

Vulnerability Detail

According to the documentation, the “Architecture Overview” page says,

“Note: The beneficiaries of a company must trust that the company owner will transfer enough tokens into the escrow. They can verify the balance of their company contract via the LiquiFi frontend app.”

However, even if the users verify the balance of the vesting contract, they cannot make sure there are enough funds in the contract for them to claim. Because users cannot know others’ claimable amounts without knowing their corresponding Merkle proofs, they cannot ensure the funds are enough for everyone to claim (unless the entire Merkle tree is made public).

Impact

If insufficient, the claiming will be based on a first-come, first-serve basis.

Tool used

Manual review

Recommendation

Even if the contract is frozen, the owner can include themselves in the Merkle tree to allow them to claim (almost) all the funds in the contract. Then, before anyone claims the tokens, the owner withdraws all of them by claimTokens.

Protocol Team Comment

Acknowledged

Issue L-06

Users cannot easily verify that a vesting contract is deployed from LiquiFi’s factory

Severity

Low

Vulnerability Detail

Users can verify that a vesting contract uses a verified vesting module (i.e., registered in the VestingModuleRegistry). However, they currently have no easy way to verify that the vesting contract is deployed from LiquiFi’s factory except, e.g., viewing the contract creation transaction.

Impact

Users face the risk of interacting with a modified vesting contract (but possibly with a verified module), which might include a backdoor (by the company owner) to prevent users from claiming tokens intentionally.

Tool used

Manual review

Recommendation

Suggest adding a getter function in the factory contracts to allow users to easily verify that a vesting contract is deployed from the factory.

Protocol Team Comment

Acknowledged

Issue L-07

The maxNumPayments limit in BatchPaymentsMissing Zero-address Validation

Severity

Low

Vulnerability Detail

Technically speaking, line 36 of BatchPayments should be >= instead of >.

Code Snippet

BatchPayments.sol#L36

Tool used

Manual review

Recommendation

Modify line 36 of BatchPayments to use >= instead of >.

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed

Issue L-08

Tokens can be claimed on behalf of anyone

Severity

Low

Vulnerability Detail

The claimTokens functions allow claiming the tokens of other users through the _beneficiary parameter.

Impact

While the tokens are sent to the correct address, this can lead to accounting issues for EOAs, or to issues with smart contracts that might rely on claiming the tokens themselves.

```
function claimAndDoSomething(...) {
    uint256 claimedAmount = token.balanceOf(address(this));
    vesting.claimTokens(address(this), ...);
    claimedAmount = token.balanceOf(address(this)) - claimedAmount;
    // do something with the tokens
    token.transfer(externalWallet, claimedAmount);
}
```

Code Snippet

All vesting contracts

Tool used

Manual review

Recommendation

Double-check if this functionality is desired.

Protocol Team Comment

Acknowledged

Issue I-01

Users must trust the company owner if not frozen

Severity

Informational

Vulnerability Detail

The vesting beneficiaries must trust the company owner to transfer sufficient tokens to the contract to cover all vesting plans.
Furthermore, unless the contract is frozen, the owner can withdraw all tokens at any time, update the eligible accounts (through the merkle root), or update to a new vesting module that returns different claimable amounts.

Vesting plans can also be archived at any time (even in the frozen state) which allows the beneficiaries to claim their vesting schedule only up to the archive time.

Code Snippet

BaseTokenVesting.sol#L93

Tool used

Manual review

Recommendation

Most of these trust assumptions have already been documented in the Trust & Security section.

Protocol Team Comment

Acknowledged

Issue I-02

BatchPayments.sol is only used in tests

Severity

Informational

Tool used

Manual review

Recommendation

Consider moving it to the lib/helper directory like the Token contract.

Protocol Team Comment

Acknowledged

Issue I-03

BaseTokenVesting.freezeVestingTerms can be called more than once

Severity

Informational

Tool used

Manual review

Recommendation

Consider reverting if frozen is already true.

Protocol Team Comment

Acknowledged

Issue I-04

Potentially misleading comment in TokenVesting.sol

Severity

Informational

Vulnerability Detail

Potentially misleading comment. “Owners can withdraw the remaining tokens if needed.”. The owners can also withdraw more than just the remaining tokens - they can withdraw the entire contract balance.

Code Snippet

TokenVesting#L212

Tool used

Manual review

Protocol Team Comment

Fixed

Sherlock Comment

Confirmed