QuillAudits

Audit Report
March, 2022

For

Amplify

Contents

Overview

Amplify by Ampt.Finance

Amplify Protocol provides access to high yield through a non-custodial platform, backed by real-world assets.

Scope of Audit

The scope of this audit was to analyze Amplify's smart contract's surrounding their voting mechanism for quality, security, and correctness.

Date: 16 February, 2022 - 24 February,2022

The following contracts were in scope:

Amplify Contract:
[1]https://github.com/amplify-labs/contracts/blob/main/protocol/contracts/Voting/VotingStorage.sol

[2] https://github.com/amplify-labs/contracts/tree/main/protocol/contracts/Voting

[3] https://github.com/amplify-labs/contracts/blob/main/protocol/contracts/utils/SmartWalletWhitelist.sol

Branch: Development

Commit: 9dd44ac15cb66685a5beb038065f1673bcd6a8bb

Fixed In: 1dcecca498e83cb26dba91126f9bbf52fe1e1fe

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Malicious libraries
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- ERC20 transfer() does not return boolean
- ERC20 approve() race
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis
Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis
Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption
In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Mythril, Slither, C4udit , Solhint

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
High	A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.
Medium	The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.
Low	Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.
Informational	These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	1	0
Closed	2	0	1	3

Issues Found – Code Review / Manual Testing

High severity issues

1. Total locked balance is not getting updated on withdraw

Voting.sol [#L325-327] function withdrawInternal resets the balance of the depositor's lock, but totalLocked balance does not get updated.

```
321     function withdrawInternal(address depositer) internal nonReentrant {
322         Lock storage lock = locks[depositer];
323         require(lock.end <= getBlockTimestamp(), "lock has not expired yet");
324
325         Lock memory oldLock = lock;
326         lock.amount = 0;
327         lock.end = 0;
328
```

Recommendation
Update totalLocked balance.

Status: Fixed
Commit: 9a08bcbea9b1182600a94d2a66eb3f4437b65ab8

2. Mappings were not being updated appropriately

Voting.sol [L#405-L#414] The delegateInternal function was deleting existing mappings, but failed to appropriately update mappings for the delegatee if the oldDelegatee was not equal to address(0).

```
400         if (delegatee == address(0)) {
401             uint256 delegateeIndex = delegationIndexInMap[oldDelegatee][delegator];
402
403             delete delegations[oldDelegatee][delegateeIndex];
404             delete delegationIndexInMap[oldDelegatee][delegator];
405         } else {
406             if(oldDelegatee != address(0)) {
407                 uint256 delegateeIndex = delegationIndexInMap[oldDelegatee][delegator];
408
409                 delete delegations[oldDelegatee][delegateeIndex];
410                 delete delegationIndexInMap[oldDelegatee][delegator];
411             } else {
412                 delegations[delegatee].push(delegator);
413                 delegationIndexInMap[delegatee][delegator] = delegations[delegatee].length - 1;
414             }
415         }
```

Recommendation
Update the delegatee with the delegator outside of the conditional statement on [L#406]

Status: Fixed
Commit: 1dcecca498e83cb26dba91126f9bbf52fe1e1fe8

Medium severity issues

No issues found

Low severity issues

3. Local Variable Shadowing

Voting.sol [#L108-110] function userOwnsTheLock makes use of the owner variable which shadows the owner variable in Ownable.sol. As a result the owner variable may be incorrect and lead to unintended behavior.

```
108     function userOwnsTheLock(Lock memory _lock, address owner) internal pure returns (bool) {
109         return _lock.owner == owner && _lock.delegator == address(0);
110     }
```

Recommendation
Rename the local variables that shadow other components.

Reference: https://swcregistry.io/docs/SWC-119

Status: Fixed
Commit: 1dcecca498e83cb26dba91126f9bbf52fe1e1fe8

4. Required Zero-Trust Policy

As the crucial aspect of AMPT's voting system is the offline agreement and signature of the delegator. If the attacker succeeds in convincing a delegator to sign a malformed message hash(which contains the attacker's desired parameters) with the help of phishing or social-engineering attacks, it may allow the attacker to delegate voting power, hence it is necessary and required that every delegator should follow a zero-trust policy and sign their message hashes by themselves.

Recommendation
Notify and announce the need of zero-trust policy to the delegators.

Status: Acknowledged

Informational issues

5. Use != 0 instead of > 0 for Unsigned Integer Comparison

When dealing with unsigned integer types, comparisons with != 0 are cheaper than > 0.
Voting.sol [#L95, #L100, #L238, #272, #343, #346,#396,#452,#457, #473]

Recommendation
Consider using !=0 for unsigned integer comparison.

Status: Fixed
Commit: 1dcecca498e83cb26dba91126f9bbf52fe1e1fe8

6. Don’t initialize variables with default values

Uninitialized variables are assigned with the types default values.
Explicitly initializing variables with their default value costs unnecessary gas.
Voting.sol [#150, #470]

Recommendation
Consider initializing variables without setting their default values.

Status: Fixed
Commit: 1dcecca498e83cb26dba91126f9bbf52fe1e1fe8

7. Cache array length outside of loop

Caching the array length outside a loop saves reading it on each iteration, as long as the array's length is not changed during the loop.
Voting.sol [#L100, #101]

Recommendation
Cache the array length outside of the loop

Status: Fixed
Commit: 1dcecca498e83cb26dba91126f9bbf52fe1e1fe8

Functional Testing Results

```
Voting Escrow
  constructor
    ✓ succeeds when setting amptToken instance (519ms)
    ✓ succeeds when setting admin to contructor argument (244ms)
    ✓ succeeds when setting smartChecker to contructor argument (219ms)
name, symbol, decimals
  ✓ should return correct name
  ✓ should return correct symbol
  ✓ should return correct decimals
changeSmartWalletChecker
  ✓ should fails because of wrong owner (39ms)
  ✓ should fails because of the same address
  ✓ should change smart contract instance
createLock
  ✓ should fails because of zero value
  ✓ should fails because of past date
  ✓ should fails because of exceed max cap of 4 years
  ✓ should create lock instance (62ms)
  ✓ should fails because already have a lock (61ms)
locked
  ✓ should return locked balance
increaseLockAmount
  ✓ should fails because of zero value
  ✓ should fails because lock expired
  ✓ should increase lock amount (50ms)
increaseLockTime
  ✓ should fails because time is greater than max cap of 4 years
  ✓ should fails because time is lower than end time
  ✓ should fails because lock expired
  ✓ should increase lock time (39ms)
withdraw
  ✓ should fails because lock expired
  ✓ should withdraw the locked tokens (89ms)
depositFor
  ✓ should fails because of zero value
  ✓ should fails because no lock created (130ms)
  ✓ should fails because lock expired
  ✓ should deposit for other user (352ms)
deletegate
  ✓ should fails because of missing lock amount
  ✓ should delegate vote for other user (231ms)
  ✓ should not break the functionality (226ms)
  ✓ should change delegate vote for other user (280ms)
balanceOf
  ✓ should return correct balance (206ms)
balanceOf 1M
  ✓ should return correct balance
totalSupply
  ✓ should return correct votePower
  ✓ should return correct votePower for 2 locks (80ms)


36 passing (16s)
```

Closing Summary

Overall several high and low level issues were found,which were fixed by the Amplify team.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the Amplify platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Amplify Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
March, 2022

For

Amplify

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com