LIDO ARAGON VOTING SMART CONTRACT AUDIT

October 01, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of LIDO. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 SECURITY ASSESSMENT METHODOLOGY

A group of auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 Project architecture review:
> Reviewing project documentation
> General code review
> Reverse research and study of the architecture of the code based on the source code only
> Mockup prototyping

Stage goal:
Building an independent view of the project's architecture and identifying logical flaws in the code.

02 Checking the code against the checklist of known vulnerabilities:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code
> Checking with static analyzers (i.e Slither, Mythril, etc.)

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the code for compliance with the desired security model:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit
> Exploits PoC development using Brownie

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of interim auditor reports into a general one:
> Cross-check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level and provide the client with an interim report.

05 Bug fixing & re-check:
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.3 PROJECT OVERVIEW

LIDO protocol is a project for stacking Ether to use it in Beacon chain. Users can deposit Ether to the Lido smart contract and receive stETH tokens in return. The stETH token balance corresponds to the amount of Beacon chain Ether that the holder could withdraw if state transitions were enabled right now in the Ethereum 2.0 network.
The Lido DAO is a Decentralized Autonomous Organization that manages the liquid staking protocol by deciding on key parameters (e.g., setting fees, assigning node operators and oracles, etc.) through the voting power of governance token (DPG) holders.
The Lido DAO is an Aragon organization. The protocol smart contracts extend AragonApp base contract and can be managed by the DAO.A smart contract tested in this audit is intended for logic to manage user voting on the launch of some scripts. But here users do not just choose `yes` or `no`. Each voice has weight. To calculate the weight of each voice, the token `MiniMeToken` is used.Below is the list the basic functions of a smart contract:

- `newVote()` - creates a new Vote
- `vote()` - makes a vote for participants
- `executeVote()` - executes a Vote
- `isForwarder()` - tells whether the Voting app is a forwarder or not
- `forward()` - creates a Vote to execute the desired action, and casts a support vote if possible
- `canForward()` - tells whether `_sender` can forward actions or not
- `canExecute()` - tells whether a Vote `_voteId` can be executed or not
- `canVote()` - tells whether `_sender` can participate in the Vote `_voteId` or not
- `getVote()` - returns all information for a Vote by its `_voteId`
- `getVoterState()` - returns the state of a voter for a given Vote by its `_voteId`
- `changeSupportRequiredPct()` - changes required support
- `changeMinAcceptQuorumPct()` - changes minimum acceptance quorum
- `unsafelyChangeVoteTime()` - changes Vote time

1.4 PROJECT DASHBOARD

Client	LIDO
Audit name	Aragon Voting
Initial version	8c46da8704d0011c42ece2896dbf4aeee069b84a
Final version	8c46da8704d0011c42ece2896dbf4aeee069b84a
Date	September 02, 2021 - October 01, 2021
Auditors engaged	3 auditors

FILES LISTING

Voting.sol	https://github.com/lidofinance/aragon-apps/blob/8c46da8704d0011c42ece2896dbf4aeee069b84a/apps/voting/contracts/Voting.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	1
Warning	4
Comment	4

CONCLUSION

Smart contracts have been audited and several suspicious places have been detected. During the audit no critical issues were found, one major, several warnings and comments were spotted. After working on the reported findings all of them were acknowledged by the client.Final commit identifier with all fixes: `8c46da8704d0011c42ece2896dbf4aeee069b84a`

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

MJR-1	Changing the shared variable can affect previous votes
File	Voting.sol
Severity	Major
Status	Acknowledged

DESCRIPTION

At the lines
Voting.sol#L126-L133 in the `unsafelyChangeVoteTime()` method is a change in the common for all voting variable `voteTime`.
Changing this variable will extend or shorten the voting time on existing voting.
This will affect the voting results, because the process will not go as planned when creating a vote.
So this action is potentially dangerous and may bring the unexpected side effects.

RECOMMENDATION

It is recommended to save the value of the variable `voteTime` in the structure `Vote`.
This will be done in the same way as for the `supportRequiredPct` and `minAcceptQuorumPct` variables.

CLIENT'S COMMENTARY

The issue is acknowledged. While changing voting duration by the code in question would affect all currently running votes, that's something which we may and will tackle with operations. The alternative would be to change the Aragon Voting smart contract code considerably, potentially introducing other issues to DAO government process.

The ops plan is outlined in Lido Improvement Proposal 4:
lip-4.md

2.3 WARNING

WRN-1	Incorrect condition
File	Voting.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines
Voting.sol#L382-L384 in the function `_canExecute()` condition is made:

```
        if (vote_.executed) {
            return false;
        }
```

At the lines
Voting.sol#L392-L394 in the function `_canExecute()` condition is made:

```
        if (_isVoteOpen(vote_)) {
            return false;
        }
```

This is equivalent to this code:

```
        if (getTimestamp64() < vote_.startDate.add(voteTime) && !vote_.executed) {
            return false;
        }
```

Thus, line 383 returns `false` for `vote_.executed`.
And on line 393 the value `false` is returned with `!vote_.executed`.

On Lines 271-272 values are written:

```
        open = _isVoteOpen(vote_);
        executed = vote_.executed;
```

This is equivalent to this code:

```
        open = getTimestamp64() < vote_.startDate.add(voteTime) && !vote_.executed;
        executed = vote_.executed;
```

So on line 271 the value `!vote_.executed` is written.
And on line 272 the value `vote_.executed` is written.

RECOMMENDATION

It is recommended to remove the `&&!vote_.executed` check in the `_isVoteOpen()` function.

CLIENT'S COMMENTARY

The check is redundant in `_canExecute` function, though it increases the readability of the method.

WRN-2	No variable check in initialize function
File	Voting.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

There is no check of the variable in the initialize function at the line

- Voting.sol#L87.
  The `token` variable is initialized in only one place.
  If the value of the address variable `token` is equal to zero when the `initialize()` function is executed, then this smart contract will have to be reinstalled.

RECOMMENDATION

It is recommended to add a check.

CLIENT'S COMMENTARY

The Aragon Voting contract is already deployed, and the value of `token` variable is correctly set to `0x5a98fcbea516cf06857215779fd812ca3bef1b32`

WRN-3	Function parameter not used
File	Voting.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the line
Voting.sol#L211, the `canForward()` function has 2 parameters. But the second parameter has no name and is not used anywhere.
In the `IForwarder` interface, the second parameter is called `evmCallScript`.
Thus, in a function from the interface, control must be delegated to the `_sender` address to execute the script.
But in fact, in the `Voting` contract, this function does not modify the data, but only outputs a boolean value with the result of checking the rights to create new votes for the address.
The second variable is not needed in the `Voting` contract.

RECOMMENDATION

It is recommended to remove the second variable for the `canForward()` function.

CLIENT'S COMMENTARY

The parameter is required for the Voting contract to conform to the Aragon `IForwarder` interface, so the function shouldn't be changed.

WRN-4	Possibility of gas overuse
File	Voting.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

At the lines
Voting.sol#L331-L341
if `voter` previosly voted `yea` and now voting `yea` again `vote_.yea` calculates twice: first it decreases and then it increases.
And the same case occurs if `voter` previosly voted `nay` and now voting `nay`.

RECOMMENDATION

It is recommended to add checking if `voter` previosly vote equals his curent vote.

CLIENT'S COMMENTARY

The code has two side-effects, which should happen even if the vote isn't changing during the call:

1. emitting `CastVote` event;
2. executing the vote if it can be executed already and `_executesIfDecided` is `true`.

2.4 COMMENT

CMT-1	Duplicate code
File	Voting.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the lines
Voting.sol#L84-L85 and 88, the variables are checked and initialized in the `initialize()` function.
Lines Voting.sol#L101-L103 do the same in the `changeSupportRequiredPct()` function.
This duplicate code can be moved into a separate internal function.

RECOMMENDATION

It is recommended to make refactoring of the source code.

CLIENT'S COMMENTARY

The error messages differ between the `initialize` and `changeSupportRequiredPct` functions, that makes the debugging easier. Having both checks in place arguably makes the code more explicit as well.

CMT-2	The source code intended for deployment should not contain `TODO` comments
File	Voting.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line
Voting.sol#L369 contains a comment with `TODO`.
But the deployed code needs to be improved, since it cannot be changed. There is no point in such comments.

RECOMMENDATION

It is recommended to delete these comment.

CLIENT'S COMMENTARY

The comment was left as-is to have minimal diff with the current Voting contract.

CMT-3	Using function without any logic
File	Voting.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At `Voting.sol` using empty useless function at Voting.sol#L191.

RECOMMENDATION

It is recommended to comment it or delete.

CLIENT'S COMMENTARY

The function `isForwarder` is required for `IForwarder` interface conformance.

CMT-4	Using unsuitable visibility modifier
File	Voting.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

Visibility of a contract functions at:

- Voting.sol#L200
- Voting.sol#L224
- Voting.sol#L234
- Voting.sol#L252
- Voting.sol#L288
  which are not called internally from the same contract should be changed to `external` to save gas.

RECOMMENDATION

It is recommended to change `public` to `external`.

CLIENT'S COMMENTARY

Functions `canExecute`, `canVote`, `getVote`, `getVoterState` could be made `external`, but were decided to be left as-is to have minimal diff with the current Voting contract. `forward` is `public` in `IForwarder` interface the Voting contract conforms to. IForwarder.sol

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

- Ethereum
- Cosmos
- EOS
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- C++

CONTACTS

https://github.com/mixbytes/audits_public

https://mixbytes.io/

hello@mixbytes.io

https://t.me/MixBytes

https://twitter.com/mixbytes