MixBytes()

Open Enterprise DotVoting Smart Contract Audit Report

Introduction

General provisions

Aragon is software allowing to freely organize and collaborate without borders or intermediaries. Create global, bureaucracy-free organizations, companies, and communities.

Autark is an Aragon Network organization building open source tools that serve digital cooperatives and aims to revolutionize work by leveraging the corresponding challenges.

With this in mind, MixBytes team was willing to contribute to Aragon ecosystem development by providing security assessment of the Open Enterprise Suite smart contracts created by Autark, as well as the StandardBounties and AragonApp smart contracts.

Scope of the audit

Code written by: Autark

Audited commit: DotVoting.sol version efd5cde.

Security Assessment Principles

Classification of Issues

- CRITICAL: Bugs that enable theft of ether/tokens, lock access to funds without possibility to restore it, or lead to any other loss of ether/tokens to be transferred to any party (for example, dividends).
- MAJOR: Bugs that can trigger a contract failure, with further recovery only possible through manual modification of the contract state or contract replacement altogether.
- WARNINGS: Bugs that can break the intended contract logic or enable a DoS attack on the contract.
- COMMENTS: All other issues and recommendations.

Security Assessment Methodology

The audit was performed with triple redundancy by three auditors.

Stages of the audit were as follows:

- “Blind” manual check of the code and model behind the code
- “Guided” manual check of the code
- Check of adherence of the code to requirements of the client
- Automated security analysis using internal solidity security checker
- Automated security analysis using public analysers
- Manual by-checklist inspection of the system
- Discussion and merge of independent audit results
- Report execution

Detected Issues

CRITICAL

Not found

MAJOR

Not found

WARNINGS

1. ADynamicForwarder.sol#L344

ADynamicForwarder.sol#L474

ADynamicForwarder.sol#L328

ADynamicForwarder.sol#L454

ADynamicForwarder.sol#L460

During the `copy` function execution (see here) extra 32 bytes are being copied. The copy function is probably being used incorrectly or contains an error. We recommend checking this behavior.

Fixed at 5490732

2. DotVoting.sol#L157

DotVoting.sol#L277

DotVoting.sol#L265

DotVoting.sol#L278

DotVoting.sol#L310

DotVoting.sol#L332

DotVoting.sol#L342

DotVoting.sol#L352

DotVoting.sol#L419

DotVoting.sol#L465

DotVoting.sol#L486

Access to a non-existent voting is allowed. We recommend adding a check that the voting id passed in the parameters exists.

Fixed at b5dd6c3

3. DotVoting.sol#L442

DotVoting.sol#L452

It is allowed to go beyond the boundaries of the `cKeys` array. We recommend adding a check that `i` does not go beyond the boundaries of the array.

Fixed at b5dd6c3

4. ScriptHelpers.sol#L74

ScriptHelpers.sol#L80

ScriptHelpers.sol#L86

ScriptHelpers.sol#L95

ScriptHelpers.sol#L50

ADynamicForwarder.sol#L456

We suggest controlling and preventing memory references from going beyond array boundaries for functions that deal directly with array memory. This will prevent errors at an early stage and reduce the risk of hard-to-diagnose memory corruption errors.

Acknowledged

COMMENTS

1. DotVoting.sol#L213

The account with the `ADD_CANDIDATES_ROLE` rights is able to block the vote. It can either add a large number of candidates or a long `_metadata` , so that further processing (`_executeVote` in particular) will be impossible due to block gas restrictions.

Acknowledged

2. DotVoting.sol#L134

DotVoting.sol#L143

DotVoting.sol#L154

DotVoting.sol#L264

DotVoting.sol#L277

DotVoting.sol#L296

DotVoting.sol#L330

DotVoting.sol#L341

DotVoting.sol#L351

We recommend adding the `isInitialized` modifier.

Fixed at b5dd6c3

3. DotVoting.sol#L159-L160

Access to a non-existent option is allowed. We recommend adding a check that `_candidateIndex` is valid.

Fixed at b5dd6c3

4. DotVoting.sol#L443

DotVoting.sol#L446

DotVoting.sol#L454

During the function execution, the value of `voteInstance.totalParticipation` can be maintained in a local variable and then written to storage at the end of the method to save gas.

Fixed at b5dd6c3

5. DotVoting.sol#L504

There is no need to copy `Action` into memory and waste gas on reading the entire structure and allocating memory. We recommend replacing the `memory` qualifier with `storage` .

Fixed at b5dd6c3

6. DotVoting.sol#L77-L80

In fact, there's no truncation of the data specified in the comment. We recommend updating the comment.

Acknowledged

7. DotVoting.sol#L204

The data type is not a `string` but an `address` . Moreover, this field is used to generate internal keys in the code, and options with the same `_description` are not allowed. We suggest verifying that this behavior is appropriate and update the comment accordingly.

*Fixed at 62bf419

8. DotVoting.sol#L53

ADynamicForwarder.sol#L57-L59

Since explicit positions of the storage data are not used, migration of the current contract instance to a new one may be complicated. A simple example of explicit storage data positions can be seen here.

Acknowledged

9. DotVoting.sol#L501

The incorrect comment was most likely copied from the function below. We recommend updating the comment.

Fixed at 62bf419

10. DotVoting.sol#L212

A check proving that the vote is still open can be added. Otherwise, it makes no sense to write data to the blockchain.

Fixed at 62bf419

11. DotVoting.sol#L376

The “Description” parameter in the comment is missing. We suggest adding it.

Fixed at 62bf419

12. DotVoting.sol#L415

We recommend checking that the length of the `_supports` array does not exceed the number of voting options.

Acknowledged

13. DotVoting.sol#L372

At the moment, the first parameter can only be an address. We recommend correcting the comment.

Fixed at 62bf419

14. DotVoting.sol#L474

ADynamicForwarder.sol#L165

ADynamicForwarder.sol#L177

ADynamicForwarder.sol#L401

ADynamicForwarder.sol#L460

ADynamicForwarder.sol#L499

We recommend using the `SafeMath` library for performing subtraction.

Acknowledged

15. ADynamicForwarder.sol#L115

During processing of the specified expression, a value truncation may occur. Voting options must not exceed 256, which is not controlled. However, the `keyArrayIndex` field is not used. We recommend either deleting the field or adding the according preliminary overflow check.

Acknowledged

16. ADynamicForwarder.sol#L44

ADynamicForwarder.sol#L45

ADynamicForwarder.sol#L57

ADynamicForwarder.sol#L120

Intermediate hashing of options can be omitted. In `Action.optionKeys` you can immediately write the addresses. Then, `optionAddresses` can be omitted. `Action.options` can have the `mapping (address => OptionState)` type.

Acknowledged

17. ADynamicForwarder.sol#L75

ADynamicForwarder.sol#L91

ADynamicForwarder.sol#L107

Access to a non-existent `Action` is allowed. We recommend adding a check that `_actionId` is valid.

Acknowledged

18. ADynamicForwarder.sol#L76

Access to a non-existent `OptionState` is allowed. We recommend adding a check that `_optionIndex` is valid.

Acknowledged

19. ADynamicForwarder.sol#L61

We recommend adding the parameters for description and additional identifiers to the event to make sure there were no errors in the script when creating the vote.

Acknowledged

20. ADynamicForwarder.sol#L122

This operation does not change the `actionInstance.optionKeys` value and may consume gas. We recommend removing the assignment.

Acknowledged

21. ADynamicForwarder.sol#L41-L42

These fields are assigned but are not used further on.

Acknowledged

22. ADynamicForwarder.sol#L414

There is no need to allocate 32 memory bytes as the value will be replaced in the next line.

Acknowledged

23. ADynamicForwarder.sol#L363

ADynamicForwarder.sol#L380

The incorrect comment was most likely copied from the function below. We recommend updating the comment.

Acknowledged

24. ADynamicForwarder.sol#L460

In case numerical values (for instance, `288` and `256`) are calculated in a sophisticated way, we do not recommend writing them into the code in a pre-calculated form. This greatly complicates the code maintainability and readability. Structural parameter changes will entail verification and/or recalculation of such values. Some of these recalculations may be missed by mistake. We recommend calculating them in the code explicitly, based on the number and nature of the calldata parameters. This will lead to higher gas consumption, but reduce the likelihood of errors.

Acknowledged

25. ADynamicForwarder.sol#L222

ADynamicForwarder.sol#L184

We recommend adding additional checks: -- the size of all arrays is the same -- no exceeding the boundaries of `infoString`

Acknowledged

CONCLUSION

The level of contract security is high, and a rather difficult task was solved by limited Solidity means.

Please note that `ADynamicForwarder` contains several different incompatible offset types: calldata offset, `bytes` array offset, EVM memory offset. They are often used together and it is difficult to distinguish the offset type of a variable, and which function receives and returns this or that offset type. Unfortunately, Solidity does not allow for the derived types (in other languages, you can introduce derivatives of `uint` types and make implicit casts impossible, thereby preventing confusion and errors). As an alternative, we suggest at least explicitly describing in the documentation, variable names and parameters which offset types they should contain.

Also, we recommend controlling and preventing accessing the memory outside of array boundaries and accessing non-existent mapping elements.

The fixed contract doesn’t have any vulnerabilities according to our analysis.