mixbytes / audits_public Public

Code Issues Pull requests 3 Actions Projects Security Insights

master

audits_public / Aragon / Open Enterprise / AddressBook.md

VadimBuyanov Adding reports ✕ History

1 contributor

135 lines (67 sloc) | 7.66 KB

MixBytes()

Open Enterprise AddressBook 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: AddressBook.sol version 66a8515.

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

Not found

COMMENTS

1. AddressBook.sol#L103

We recommend adding the explicit check `isInitialized`.

Fixed at ed2f199

2. AddressBook.sol#L91

There is a constant for this kind of error message - `ERROR_CID_MALFORMED`. We recommend factoring out the entire check as a modifier.

Fixed at ed2f199

3. AddressBook.sol#L33

There is no way to get the entire list of addresses stored in the address book. An array can be added to keep track of all present addresses. When an entry is deleted from the address book, the last array element can replace the deleted element to prevent array fragmentation.

Fixed at ed2f199

4. AddressBook.sol#L33

Explicit positions of the storage data are not used, that can complicate migration of the current contract instance to a new one. A simple example of explicit storage data positions can be seen here. The important thing to understand is that the memory layout of the next version has to match the memory layout of the current version exactly. Otherwise, the following code version will have troubles accessing the data stored in the proxy storage. E.g., any addition of a base contract will ruin the layout, as it will shift the offsets of the fields.

Acknowledged

5. AddressBook.sol#L62

It is expected that structured content objects for the entries will be stored in IPFS. Users and developers should keep in mind that IPFS does not guarantee data availability. After some time unused data is removed from IPFS unless explicitly pinned by some node.

Acknowledged, dev notes were created at AddressBook.sol#L60

6. AddressBook.sol#L64

IPFS addresses have the form of `<encoding>.encode(multihash(<digest>, <function>))` (https://github.com/multiformats/multihash#example), that makes the check of line 64 valid only for base58 encoding of sha256 hashes.

Acknowledged, comments about supported encoding are at https://github.com/AutarkLabs/planning-suite/tree/ed2f199ddda280d1e7033648b69399547f05eec7

7. AddressBook.sol#L86

Similarly to the `_cid` argument of the `removeEntry` function, an additional argument `oldCid` and a content check can be introduced to prevent race conditions and ensure that the updated entry was to be updated.

Fixed at ed2f199

8. AddressBook.sol#L62

AddressBook.sol#L75

AddressBook.sol#L86

The functions can be marked as `external` to save some gas.

Fixed at ed2f199

9. AddressBook.sol#L33

IPFS-address can be stored as an array of bytes instead of a string. A more “smart” check may be introduced for the adding/updating code (to dynamically identify the hash function used and the necessary input size).

Acknowledged

CONCLUSION

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