MixBytes()

Standard Bounties Smart Contract Review

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: Bounties Network

Reviewed commit: Standard Bounties version e79d844.

Detected Issues

CRITICAL

1. StandardBounties.sol#L381

Contributions are not tagged as `refunded`. Funds can be re-withdrawn from the bounty balance by the `refundContribution` requests from contributors who have already received funds during the `refundContributions` call.

An example of the attack vector:

- A contributor `C1` makes a contribution of 10 Ether with contribution id = 0.
- A contributor `C2` makes a contribution of 10 Ether with contribution id = 1.
- An issuer `I1` that does not relate in any way to `C1` or `C2` issues a `refundContributions` call with the `_contributionIds` parameter being equal `[1]`.
- As a result of this call, `C2` receives 10 Ether back. After the call, the balance of the bounty is 10 Ether.
- `C2` issues a `refundContribution` call with the `_contributionId` parameter being equal to `1`. As a result of this call, `C2` receives the remaining 10 Ether.

Ultimately, `C2` received a double refund, and the contribution of `C1` was in fact transferred to `C2`. The balance of the bounty is 0, despite the fact that the issuer intended to refund only `C2`.

Fixed at StandardBounties.sol#L385

MAJOR

Not found

WARNINGS

1. StandardBounties.sol#L237

The function returns nothing, although the function signature indicates that the function should return a `uint`.

Fixed at StandardBounties.sol#L254

2. StandardBounties.sol#L407

If an issuer does not withdraw all funds from the bounty, this may cause inequality between contributors. Some of them may manage to withdraw the remaining funds, and others do not.

Acknowledged

StandardBounties Team: this function is meant to be used as a fail safe, with maximum flexibility to allow the issuers to withdraw funds in whatever quantity they please. I've added a comment to this function as a warning to others who use it, to watch out for this edge-case.

3. StandardBounties.sol#L376

The `_contributionIds[i]` is allowed to go beyond the `bounties[_bountyId].contributions` array bounds. We recommend replacing comparison with the strict one.

Fixed at StandardBounties.sol#L378

4. StandardBounties.sol#L372 ,

StandardBounties.sol#L402 ,

StandardBounties.sol#L506 ,

StandardBounties.sol#L639

As going out of array bounds in these functions is not controlled, `metaTxRelayer` can pass the `0` address as `sender` and successfully pass the access checks. Thus, an attacker who gained access to `metaTxRelayer` can bypass access control in some cases.

As transaction relayer is out of the audit scope, it is not possible to assess the security risk in this case. However, we recommend checking the digital signature of the account involving the relay transaction either in the contract code or in the Transaction relayer code. Function selector and all function parameters must be signed with a digital signature.

Acknowledged

StandardBounties Team: yes this is absolutely correct - the meta transaction relayer indeed checks signatures and is deployed at the same time as the main contract (before it begins being used) so that it cannot be captured by any 3rd party attackers.

COMMENTS

1. StandardBounties.sol#L19

Gas consumption can be optimized

- `deadline - uint64` may be used for timestamp
- `tokenVersion - uint8` may be used (with `0` constant for Ether, `1` for ERC-20, `2` - for ERC-721).
- `deadline`, `hasPaidOut` and `tokenVersion` should be placed after `token` to use one storage slot for these fields

2. StandardBounties.sol#L60

Gas consumption can be reduced by using ReentrancyGuard

Explanation can be found here and here.

3. StandardBounties.sol#L303 ,

StandardBounties.sol#L816

Due to the check at the stage of bounty creation, `tokenVersion` can have only `0` , `20` , `721` values. If there are no logical errors in the contract, control never reaches the given code lines. We recommend using `assert` instead of `revert` for checking code consistency.

4. StandardBounties.sol#L275

We recommend adding a check that the deadline has not passed. Otherwise, contribution is meaningless as such.

5. StandardBounties.sol#L198

We recommend adding a check that the deadline has not passed.

6. StandardBounties.sol#L111 ,

StandardBounties.sol#L120 ,

StandardBounties.sol#L130 ,

StandardBounties.sol#L140

We recommend adding checks that the specified bounty exists and the array bounds are not exceeded in this modifier and alike. Otherwise, access control modifiers will be satisfied when passing `_sender` equal to `0` .

In some functions, array bound excess is checked separately, in others it is not checked at all. In any case, we believe that these checks should be in access control modifiers.

CONCLUSION

Overall code quality is rather high. However, there are some flaws, including the critical one, which was successfully fixed by the original contract authors, the Bounties Network, with a new contract version deployed.