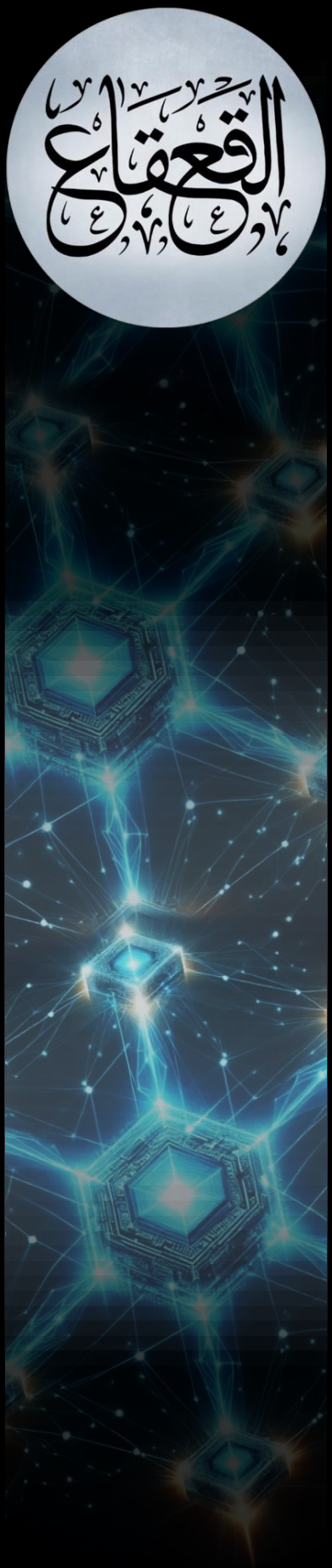
Al-Qa'qa’



Merit Systems
Auditing Report

Auditor: Al-Qa'qa’

18 June 2025

Table of Contents

1 Introduction

1.1 About Al-Qa'qa'

Al-Qa'qa' is an independent Web3 security researcher specializing in smart contract audits. Success in placing top 5 in multiple contests on Cantina and Sherlock. In addition to smart contract audits, he has moderate experience in core EVM architecture, geth.

For security consulting, reach out to him on Twitter - @Al_Qa_qa

1.2 About Merit Systems

Merit Systems enable direct monetization of GitHub repos. They create repo-owned bank accounts, simple financial tools for monetization, and automatic impact-weighted payouts to contributors.

1.3 Disclaimer

Security review cannot guarantee `100%` the protocol's safety. In the Auditing process, we try to identify all possible issues, and we cannot be sure if we missed something.

Al-Qa'qa' is not responsible for any misbehavior, bugs, or exploits affecting the audited code or any part of the deployment phase.

And change to the code after the mitigation process, puts the protocol at risk, and should be audited again.

1.4 Risk Classification

Severity	Impact:High	Impact:Medium	Impact:Low
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

1.4.1 Impact

- High - Funds are **directly** at risk, or a **severe** disruption of the protocol's core functionality
- Medium - Funds are **indirectly** at risk, or **some** disruption of the protocol's functionality
- Low - Funds are **not** at risk

1.4.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align or little-to-no incentive

2 Executive Summary

2.1 Overview

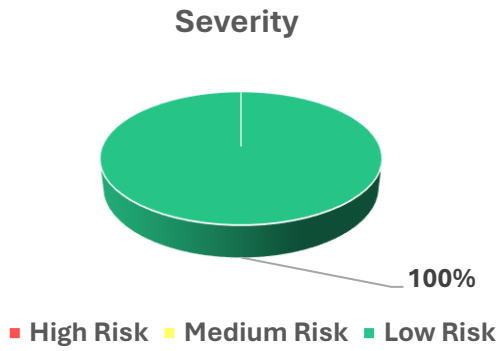
Project	Merit Systems
Repository	Ledger (Private)
Commit Hash	86786d6c6728fad690be7e2c7a76640f6ad7d0b8
Mitigation Hash	11c38224800bf0304f98bc85c257cffaa3caf442
Audit Timeline	13 June 2025 to 15 June 2025

2.2 Scope

- src/Payments/Escrow.sol
- script/Deploy.Base.s.sol

2.3 Issues Found

Severity	Count
High Risk	0
Medium Risk	0
Low Risk	3



3 Findings Summary

ID	Title	Status
L-01	alaising `uint` is incompatible with EIP712	Fixed
L-02	Incorrect even emition at `reclaimRepoDistributions()`	Fixed
L-03	At `timestamp` equals `claimDeadline` both claiming and reclaiming is allowed after instant reclaimability support	Fixed
I-01	Incorrect construction of `CLAIM_TYPEHASH` because of wrong deadline parameter naming	Fixed
I-02	empty distributions are not checked when doing Escrow operations	Fixed

4 Findings

4.1 Low Findings

4.1.1 alaising `uint` is incompatible with EIP712

context: Escrow.sol#L25-L28

Description
When constructing the type hash of the EIP712 of the signature schema, we use `uint` instead of `uint256` when constructing the hash

Escrow.sol#L25-L28

```
    bytes32 public constant SET_ADMIN_TYPEHASH =
        keccak256("SetAdmin(uint repoId,uint accountId,address[] admins,uint
nonce,uint signatureDeadline)");
    bytes32 public constant CLAIM_TYPEHASH =
        keccak256("Claim(uint[] distributionIds,address recipient,uint nonce,uint
deadline)");
```

This violates theEIP712 as it do not use alaising `uint` and `int`

https://eips.ethereum.org/EIPS/eip-712#definition-of-typed-structured-data-𝕊

> Definition: The atomic types are `bytes1` to `bytes32`, `uint8` to `uint256`, `int8` to `int256`, `bool` and `address`. These correspond to their definition in Solidity. Note that there are no aliases `uint` and `int`. Note that contract addresses are always plain address. Fixed point numbers are not supported by the standard. Future versions of this standard may add new atomic types.
>
> This can result in inability for some wallets to decode the signature and output a human readable info for users.

Recommendations

We should not use alaisings like `uint` or `int` and use `uint256` and `int256` instead. Here are the affected hashes:

- SET_ADMIN_TYPEHASH
- CLAIM_TYPEHASH
- _domainSeparator()::EIP712Domain

Status: Fixed at PR-120

4.1.2 Incorrect even emition at `reclaimRepoDistributions()`

context: Escrow.sol#L430

Description

When reclaiming the repo distributions, it is be design to allow anyone to reclaim the distributions. But the even emitted when reclaiming a given repo `ReclaimedRepoDistribution`, is putting `msg.sender` in the place of the `admin` of the event.

Escrow.sol#L430

```
    function reclaimRepoDistributions( ... ) external {
        ...
        for (uint i; i < distributionIds.length; ++i) {
            ...

>>          emit ReclaimedRepoDistribution(batchId, distributionId, msg.sender,
distribution.amount);
        }
        emit ReclaimedRepoDistributionsBatch(batchId, repoId, accountId,
distributionIds, data);
    }
// --------------
    // interface/IEscrow.sol
    event ReclaimedRepoDistribution(
        uint256 indexed batchId,
        uint256 indexed distributionId,
>>      address indexed admin,
        uint256 amount
    );
```

Since anyone can reclaim Repo distributions, the `msg.sender` may be one of the admins, or any other user. So, in case non-admin sender reclaimed, the event will get fired putting the reclaimer in the admin slot and he is not one of the admins for that `repo/account` account.

Recommendations

We can change the event to put `reclaimer` instead of `admin` to make the event emit correctly.

Status: Fixed at PR-121

4.1.3 At `timestamp` equals `claimDeadline` both claiming and reclaiming is allowed after instant reclaiming support

context: Escrow.sol#L359

Description

After supporting instant reclaimability support, Admins can reclaim at the same timestamp of creating the distribution. To allow this we changed the check of reclaiming from `>` to `>=`.

The problem is that this makes the two states `claim` and `reclaim` are allowed
when `timestamp` equals `claimDeadline` as claiming is allowed
when `timestamp` equals `claimDeadline` and also reclaiming is allowed
when `timestamp` equals `claimDeadline`.

Escrow.sol#L358 | Escrow.sol#L424

```
    function claim( ... ) external {
        ...
        for (uint i; i < distributionIds.length; ++i) {
            ...
>>          require(block.timestamp        <= distribution.claimDeadline,
Errors.CLAIM_DEADLINE_PASSED);
            ...
        }
        emit ClaimedBatch(batchId, distributionIds, msg.sender, data);
    }
// ------------------
    function reclaimRepoDistributions( ... ) external {
        ...
        for (uint i; i < distributionIds.length; ++i) {
            ...
>>          require(block.timestamp     >= distribution.claimDeadline,
Errors.STILL_CLAIMABLE);
            ...
        }
        emit ReclaimedRepoDistributionsBatch(batchId, repoId, accountId,
distributionIds, data);
    }
```

Since both checks use `or equal` at a timestamp equal to `claimDeadline`, claiming and reclaiming checks can pass. This can result in one front running the other. recipient claim, but Admins reclaimed before he redeemed his signature at the last second.

Recommendations

Since supporting instant reclaiming we should use `>=`. we can use `<` instead of `<=` in `claim()` so that the contract can't reach a state where at a given time, both claiming and reclaiming are allowed.

Status: Fixed at PR-122 by changing the design so you can claim, as long as it was not reclaimed

4.2 Informational Findings

4.2.1 Incorrect construction of CLAIM_TYPEHASH because of wrong deadline parameter naming

context: Escrow.sol#L27-L28

Description

When constructing `CLAIM_TYPEHASH` we use `deadline` naming for the deadline parameter. However, this is not the parameter name of the function `claim()` where the name of the parameter is `signatureDeadline` and not `deadline

Escrow.sol#L27-L28

```
    bytes32 public constant CLAIM_TYPEHASH =
        keccak256("Claim(uint[] distributionIds,address recipient,uint nonce,uint
deadline)");
// ----------
    function claim(
        uint[] memory  distributionIds,
>>      uint256        signatureDeadline,
        uint8          v,
        bytes32        r,
        bytes32        s,
        bytes calldata data
    ) external { ... }
```

It is better to not mismatch between TypeHash constructed with the parameter naming used in it, for consistency and sticking with EIP712.

Recommendations

We can change `deadline` to `signatureDeadline` in `CLAIM_TYPEHASH`

Status: Fixed at PR-123

4.2.2 empty distributions are not checked when doing Escrow operations

context:

- Escrow.sol#L205
- Escrow.sol#L254
- Escrow.sol#L405
- Escrow.sol#L438

Description

When handling distributions, we are not checking for empty (zero length array) distributions. This include distrubuting functions (Solo/Repo) and reclaiming functions (Solo/Repo).

Since we are not enforcing the length to be greater than zero, calling these functions will result in increasing `batchCount` and emit Batch events (Distrubute/Reclaim batch events). without even a single distributions created or reclaimed.

Recommendations

We can enforce array length input greater than zero for Distribution creation and reclaiming

Status: Fixed at PR-124