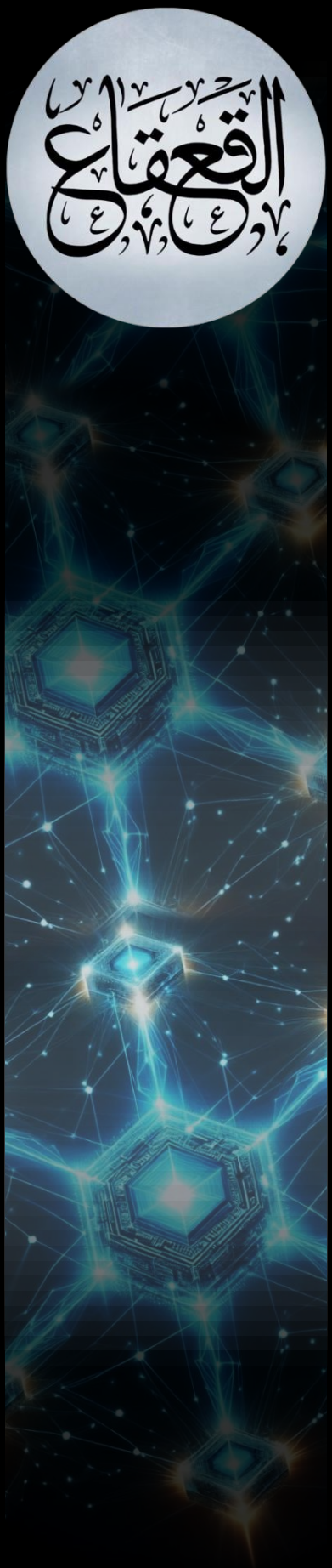
Al-Qa'qa’



Merit Systems

Auditing Report

Auditor: Al-Qa'qa’

20 April 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

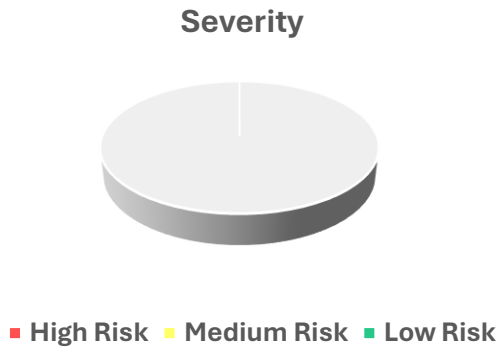
Project	Venice
Repository	Ledger (Private)
Commit Hash	907bff682a5cdd36e4e43386fe99c33aae516048
Mitigation Hash	8f2c5e765fbc6bea6abc596e62e4cfed94dcfe87
Audit Timeline	19 Apr 2025

2.2 Scope

- src/Payments/Escrow.sol

2.3 Issues Found

Severity	Count
High Risk	0
Medium Risk	0
Low Risk	0



3 Findings Summary

ID	Title	Status
I-01	Batch count can be incremented with empty deposits	Fixed

4 Findings

4.1 Informational Findings

4.1.1 Batch count can be incremented with empty deposits

Description

`batchDeposit()` function is not enforcing a minimum length of the deposits. It can accept `0` length array of `DepositParams`. This will allow the function to success without taking any funds or making an Escrow Deposit to the user.

```
    function batchDeposit( ... )  ... {
        require(params.length <= batchDepositLimit,
Errors.BATCH_DEPOSIT_LIMIT_EXCEEDED);
        depositIds = new uint[](params.length);

        for (uint256 i = 0; i < params.length; i++) {
            depositIds[i] = deposit(params[i]);
        }

        emit BatchDeposited(batchCount++, depositIds, data);
    }
```

This should be OK and has no Impact. as since `batchCount` is of `uint256` type, and we are on Base where Fees are too low. He can't spam too much requests till reaching the max of the variable. it is just a functionality works incorrectly.

Recommendations

Check that the length of Patch is greater than `0`.

Status: Fixed