ILLUSTRATIVE SAMPLE · A FICTIONAL SYSTEM, NOT A CLIENT'S

Payments and ledger integrity check: the rewards app

Prepared by John Karanja · fixed scope · Sample, November 2026

One payout flow in a rewards app that pays players small amounts of cryptocurrency for correct answers. It runs from the player's answer, through the score service and the reward claim, to the payout sender and the chain. Traced on a sample of records and logs with private details removed; no live transactions, logins or access to funds. This is not a financial audit, a smart-contract audit, or a tax or accounting opinion.

Summary

When nothing goes wrong, the flow works. The sample holds 1,240 payout rows from 6 to 12 October. Every row marked sent has a matching transfer on the chain. The weak point is a chain that does not answer in time. The sender treats this timeout as a failure and sends again. Nothing in the records shows the request was already claimed. The sender log shows 37 timeouts that week. For 4 of them, the chain shows two transfers of the same amount to the same wallet. The balance a player sees is also kept in two places, and nobody compares them. The first fix is small and comes before anything else: claim each request before sending it.

> Sound: each confirmed payout matches one row in the payout table, and the scoring rules are well tested.

> Fragile: a timeout triggers an unchecked retry, which can pay the same request twice.

> Do first: a claim record with a unique request id, checked before every send.

Scope and method

In scope: the payout flow from answer to chain, the payout table, the reward balance held by the score service, the sender's retry logic and the operations guide. Not in scope: smart contracts, who holds the wallet keys, in-app purchases, security testing, financial audit, tax and accounting.

Materials reviewed:

- Read-only access to the code of the score service and payout sender, as it was on 14 October.
- An export of the payout table for 6 to 12 October, with private details removed: 1,240 rows, 1,212 marked sent and 28 failed.
- The payout sender log for the same week.

- Public chain records for the app's sending wallet over the same week.
- The four-page operations guide and a short architecture note from the founding developer.
- Two calls with the founder and the developer.

Evidence has three labels. Observed: I saw it in the materials, and I name the source. Reported: the team told me, and I did not check it. Unknown: the evidence was not available. I never count an unknown as zero.

The map

1 **Player answers** The app records each answer and sends it to the score service with the player and question ids.

2 **Score service** 3 It scores answers and keeps its own running reward balance for each player.

3 **Reward claim** When a player taps claim, a row is written to the payout table with the amount and wallet address, marked pending.

4 **Payout sender** 1 Every 15 seconds, a worker picks up pending rows. It sends the transfer without first recording that it has claimed the row.

5 **The chain** 2 The sender waits 30 seconds for an answer, then treats silence as failure and resends without checking.

Findings

1. A payout can be sent twice

HIGH

EVIDENCE	Observed. The sender retries on a timeout, and nothing records that a request was already claimed. A row stays pending until the chain confirms it. The sender log for 6 to 12 October shows 37 timeouts. For 4 of them, the chain records show two transfers of the same amount to the same wallet within two minutes, but only one row in the payout table.
IMPACT	Each double send is money that leaves the wallet without showing in the records. It cannot be recovered without the player's help. Timeouts happen most at busy times (31 of the 37 fell between 19:00 and 22:00), so the risk grows with use.
UNKNOWN	Nothing important.
RECOMMENDATION	Claim each request before sending. Write a claim record with a unique request id in the same database transaction that moves the row out of pending. A retry checks the claim first, and never decides to send on its own.
DONE WHEN	In a test environment, force a timeout on the chain call 20 times. The chain shows exactly one transfer per request.

2. A timed-out payout has no owner

HIGH

EVIDENCE	Observed. Timeouts are only logged as warnings; no person reviews them. The payout table has three states: pending, sent and failed. There is no state for "we do not know". The guide says to "check the logs if a player complains" and names no person.
IMPACT	A payout with an unknown result is either sent again without checking, as in finding 1, or forgotten. A player who was not paid has to notice and complain. A player paid twice is never found.
UNKNOWN	Nothing important.
RECOMMENDATION	Add an "uncertain" state. Before any retry, check the row against the chain (reconcile it). Only a row confirmed absent from the chain may be sent again. Name one person who reviews any row left uncertain for more than an hour.
DONE WHEN	Replay the 37 timeouts from 6 to 12 October through the reconciliation step. Each ends as sent or failed, with the chain record that decided it.

3. Two records hold one balance

MEDIUM

EVIDENCE	Observed. The score service and the payout table each keep a total. No job compares the two.
IMPACT	When they differ, nobody can say which is right, and a player's complaint cannot be answered from the records.
UNKNOWN	How often they have differed. The export did not include the score service balances, so no comparison was possible.
RECOMMENDATION	Pick the one record that counts; work out the other from it. Until then, compare them daily.
DONE WHEN	For a sample of 200 players, the balance on screen equals the sum of their payout rows. The daily comparison shows no unexplained difference for seven days in a row.

4. Wallet top-ups are not recorded

MEDIUM

EVIDENCE	Observed. The guide says the founding developer tops up the sending wallet "when it runs low". The chain records show three top-ups in the week reviewed. None appears in the payout table or in any other record supplied.
IMPACT	The wallet balance cannot be checked against the records. Money coming in is invisible, so a double send looks like ordinary spending.
UNKNOWN	Who else can top up or withdraw from the wallet. Who holds the keys was out of scope.
RECOMMENDATION	Record every top-up with its chain reference. Reconcile daily: opening balance, plus top-ups, minus confirmed sends, equals the closing balance on the chain.
DONE WHEN	For the week reviewed, the daily check matches the chain balance, or lists each difference with a reason.

5. The failure cases have no tests

MEDIUM

EVIDENCE	Observed. The test suite has 46 tests for the scoring rules and 3 for the payout sender, all on the success path. None covers a timeout, a duplicate claim, a rejected transfer or a restart in the middle of a send.
IMPACT	The fixes for findings 1 and 2 cannot be shown to work, and a later change could undo them without anyone noticing.
UNKNOWN	Nothing important.
RECOMMENDATION	Add tests for six failure cases: a timeout before the transfer goes out, a timeout after it, a double tap on claim, a restart between send and mark, a rejected transfer, and an empty wallet.
DONE WHEN	All six tests run on every change. Putting the unchecked retry back makes at least one of them fail.

6. A double tap can create two claims

LOW

EVIDENCE	Observed. The claim step does not check for an existing pending claim. In the export, 9 players have two claim rows written within one second. In each case the second row failed at send, with "insufficient balance" read from the score service.
IMPACT	Today the second claim fails only by luck. Fixing finding 3 would remove that protection.
UNKNOWN	Nothing important.
RECOMMENDATION	The claim button sends a request id, and the claim step refuses a second pending claim for the same player.
DONE WHEN	Two claims submitted within 100 milliseconds produce one row.

What is sound

- Confirmed payouts match. All 1,212 rows marked sent have a chain transfer of the same amount to the same wallet.
- A row is marked sent only after the chain confirms it, not when the transfer is first sent.
- The scoring rules are well tested: 46 tests with clear names, all passing at the version reviewed.
- Wallet addresses are checked at claim time, and the export shows no transfer to a badly formed address.

Action plan

First

1. Add a claim record with a unique request id.
2. The sender refuses any request without a claim.
3. Done when: a forced timeout never causes a second send.

Next

1. Add the uncertain state and the reconciliation step, and name the person who reviews uncertain rows.
2. Run the six failure tests on every change.
3. Record wallet top-ups and start the daily wallet reconciliation.

Later

1. Make the payout table the one balance that counts; work out the screen figure from it.
2. Refuse duplicate claims by request id.

3. Review the reconciliation results once a month, and decide what to do about the 4 double sends already found.

Open questions

- Can a transfer carry a unique reference on the chain in use? This decides whether reconciliation matches by request id or by wallet, amount and time.
- Who holds the sending wallet's keys, and who can move funds? Out of scope, but it matters to finding 4.
- Should the 4 double sends be recovered from players? That is a business decision, not a technical one.

Next steps

I walk you through the findings on a call, then answer one round of written questions. If you hire me for the fixes within 60 days, this check's full fee comes off their price.

This is a sample: a fictional system, written like a real report. A real report follows the scope we agree in writing.

John Wainaina Karanja · UTC+3 · info@johnkaranja.com · johnkaranja.com/book