WORKING PAPER

The Accountant Who Saw an Empty Shop: Separating Visibility From Control in Micro-Enterprise Software

August 2026 · ~3,100 words

Idarus Ali
Founder, AskBiz

ABSTRACT

Most access-control systems compress two different questions into one: how much can this person see, and how much can they change. For an owner-run micro-enterprise, that compression is a design defect, not a simplification. A hired accountant, a silent business partner, and a cashier are three different relationships, and a system that only offers “staff” or “owner” has nowhere to put the first two. This working paper reports what happened when a point-of-sale platform used by small businesses across East Africa and the Middle East tried to model those relationships properly, and what it found while doing so. The starting failure was not a locked door; it was an unlocked one that opened onto the wrong room. Any authenticated user of the platform was, until this year, silently treated as the full owner of their own account — so an accountant invited in good faith to review someone else’s business would sign in and see their own account instead: no products, no sales, nothing obviously broken enough to report. The fix separates two dimensions that most role systems bundle together. A business partner, established to function as a co-owner in daily practice, now receives full operational parity. An accountant or auditor receives a narrow, read-only grant into financial reporting alone, deliberately built as a separate, additive path so that a permission gap anywhere else in the product can never widen their reach. The paper places this design against three points of comparison — a read-only administrator role already established at Okta, an accountant and bookkeeper collaborator role already offered by Wave, and the apparent absence of any equivalent built-in role at Loyverse, a point-of-sale competitor serving the same small-retail market — and reports plainly on what is still unverified: the new grant has not yet been exercised by a real delegated account in production, and building it surfaced a second, older

population of users who had been silently affected by the same failure for longer, in an unrelated part of the same product.

KEY FINDINGS

1. Delegated access failed silently, not loudly. Before this year, any authenticated user of this product was treated as the full owner of their own account — so an accountant invited to review a real business would sign in and see their own empty one: no products, no sales, nothing that looked like an error rather than a business that simply hadn't started yet.
2. Trust is not one dial. Software access control usually assumes that more trust means more of everything — more visibility and more control together. A hired accountant needs the first without the second: full sight of the numbers, no ability to touch a price, a refund, or a sale. Modelling that split turned out to be the actual work.
3. A narrow permission is safer built twice than built wide once. Rather than loosen the product's main authorization check to let accountants and auditors through, the fix added a second, separate check used only by the one screen they are allowed to see — so a future bug anywhere else in the product cannot accidentally hand them more than that.
4. A business partner is not a bigger kind of employee. A partner who put in half the capital functions as a co-owner in daily practice, not a trusted senior member of staff — so the product now gives that role full parity with the owner, rather than a larger version of a permission ladder built for hired staff.
5. Fixing the front door found an older one left open the same way. Building this surfaced a second, unrelated population of already-invited team members who had been silently shown an empty version of someone else's business for longer than the feature described here has existed — disclosed here rather than fixed quietly.

FULL PAPER

1. Introduction

Two papers from this desk precede this one. The first, Surviving on the Margins, argued that Kenya's micro informal enterprises need security before they need formality — a safe

place for savings, a trading spot nobody can take on a bribe, income that survives a bad week. The second, The Till They Couldn't Open, reported what it took to build software that such a business could actually use at all: a camera instead of a keyboard, a phone number and a PIN instead of a password, an offline mode that behaves as the normal case rather than an emergency one.

This paper picks up after both of those. It assumes the business already exists, and that its owner has already gotten the software open. It asks a narrower question that only comes up once a business is actually running the tool day to day: who else gets to touch it, and how much of it should they be allowed to touch? An owner rarely runs a business entirely alone. There is often a hired accountant who checks the books once a month. There is sometimes a business partner who put in half the money and expects to see everything as a matter of course. There is, in family businesses especially, a relative who helps out on busy days without being on the payroll in any formal sense. None of these people is quite an employee, and none of them is the owner. Most access-control systems, including this one until recently, have nowhere to put them.

2. Employee or Owner, Nothing Between

The product already had a role in its own records literally called "accountant." It had existed for months, in a completely different part of the system — a business-intelligence dashboard, built for a company sophisticated enough to want its own reporting stack, that let an owner invite team members under labels including analyst, accountant, and buyer. None of those labels reached the point-of-sale side of the product at all. An owner running a market stall or a small shop through the till had no way to invite anyone into anything; the only two states available were "the person logged in with the owner's own credentials" and "a till cashier with a PIN," and a cashier's permissions are built for someone who rings up sales, not someone who reviews them.

This gap is not a small oversight. Put simply: a hired accountant is not a stranger and not an employee. They are trusted with the most sensitive part of the business — the actual numbers — for a few hours a month, and trusted with none of the rest of it. A business partner is a different relationship again: someone who put in real capital or does real work and expects, reasonably, to see and do everything the owner can. A system built only for "owner" and "staff" has no honest slot for either of them, and the honest slot matters, because the alternative is always the same: share the owner's own login and PIN, and hope nothing goes wrong.

3. Two Kinds of Trust

Most role-based access systems make an implicit assumption: that trust is one quantity, and a higher role simply gets more of it, in every direction at once. An “admin” role can see more and do more than a “member” role; a “manager” can see more and do more than a “cashier.” That assumption is usually harmless, because in most organisations, the person you trust to see sensitive information is also the person you would trust to act on it.

That assumption breaks for the accountant relationship specifically. An owner may trust their accountant completely with the real sales and profit numbers — more than they trust some of their own staff — while trusting them with precisely nothing operational: not a price change, not a refund, not a new sale rung up on the till. This is not partial trust. It is two different kinds of trust, aimed in different directions, and a role system that can only turn one dial up or down cannot represent it. Solving the accountant problem properly meant treating financial visibility and operational control as two separate settings rather than two points on the same scale — a distinction the previous section’s role list had a name for but no working mechanism behind.

4. The Empty Shop

Here is what the product actually did, until this year, once someone accepted an invitation into any of those roles and tried to use the point-of-sale side of the business: it authenticated them normally — the invitation and the login both worked exactly as expected — and then treated them as the full owner of their own account. Not the account of the business they had just been invited into. Their own.

For a genuine outside accountant, whose own account had never been used to run a shop, this meant landing on a point-of-sale dashboard with no products, no staff, no sales, and no transaction history — because there weren’t any. Nothing about that screen looks like an error. An empty dashboard on a product that legitimately supports brand-new, not-yet-stocked businesses is indistinguishable, at a glance, from a genuine new account. A conscientious accountant, invited by a real client, would have had every reason to conclude either that the business genuinely had no recorded activity yet, or that they themselves had done something wrong accepting the invitation — not that the software had quietly substituted their own account for the one they were meant to be looking at.

This is a worse failure than an outright access-denied message, and it is worth naming as a general pattern beyond this one product: a permissions bug that fails by showing a

plausible-looking wrong answer is far more dangerous than one that fails by refusing outright, because the first kind gives nobody a reason to report it.

5. A Grant Built Narrow, Not a Door Built Wide

The fix had two parts, and the second part is the one worth describing carefully, because it is a decision about how to build the fix, not just what the fix should do.

The first part was straightforward: before treating a logged-in user as an owner, the product now checks whether that person is actually a delegated team member on someone else's account, and if so, resolves them against the real business they were invited into rather than their own. A genuine solo owner, who has never invited anyone, is completely unaffected by this change — the check simply confirms what was already true for them.

The second part is about restraint. It would have been simpler to let accountant and auditor roles through the same general-purpose check used for every other kind of access, with a permissions list attached that happened to be narrow. That was deliberately not done. Instead, the one screen accountants and auditors are allowed to see — a read-only financial and activity report — has its own separate, additive check, one that every other part of the product ignores entirely. The practical effect is that a bug introduced later, anywhere else in a large and actively developed product, cannot accidentally widen an accountant's reach beyond that one screen, because nothing else in the product knows how to grant them anything at all. A permission is safer when it is impossible to reach from the wrong place, not merely unlikely to be reached from it.

6. The Partner Is Not a Bigger Employee

A business partner received the opposite treatment, deliberately. Where the accountant relationship called for narrowing a grant to almost nothing, the partner relationship called for granting full parity with the owner — every tab, every action, exactly as if they were the account holder themselves.

The reasoning is about what a partnership actually is in practice, not about seniority. A partner who contributed half the starting capital, or who runs the counter on alternating days, is not a highly trusted employee; in every functional sense that matters to how the business is actually operated, they are a second owner who happens to log in under a different account. Treating that relationship as an elevated staff tier — more permissions than a manager, fewer than a true owner — would have modelled a hierarchy that does not exist in the real relationship it was meant to represent. The distinction between the

accountant grant and the partner grant is the same distinction the rest of this paper has been making throughout: access should match the real relationship it stands in for, not a generic ladder of trust with more rungs added at the top.

7. Software That Has Solved Part of This

This problem is not new, and it is worth being honest about how much of it other software has already worked out, and where the pattern still seems to be missing entirely from products serving this specific market.

Okta, an identity and access-management platform built for large organisations, has long offered a dedicated read-only administrator role: full visibility into the admin console, no ability to change anything in it, defined as its own distinct role rather than a weaker version of a higher one — the accountant and auditor distinction described here, solved years earlier for a different kind of enterprise. Wave, an accounting platform aimed at small businesses, explicitly supports inviting an accountant, bookkeeper, or business partner as a named collaborator type, which confirms that the underlying relationship this paper describes is a recognised, common one, not a quirk of this product's own user base — though it is also worth noting, in the same spirit of honest disclosure as the rest of this series, that Wave moved collaborator access behind its paid plan in mid-2026, which is exactly the kind of barrier a genuinely cash-constrained micro-enterprise cannot always clear.

The pattern looks different in software built specifically for small retail. Loyverse, a point-of-sale competitor operating in the same market as this product, ships four default roles — owner, administrator, manager, cashier — with no accountant-equivalent among them; matching this paper's read-only financial grant there would mean an owner hand-assembling a custom role from a checklist of individual permissions themselves, correctly, with no guidance that accountant-style access is even a common enough need to have a name. That is a real gap in software built for exactly this population, and closing it without requiring the owner to get a permissions checklist right themselves was the actual point of the work described in this paper.

8. The Second Door

One finding here is reported specifically because it reflects badly on how long the underlying problem existed, in the same spirit of plain accounting the previous paper in this series committed to.

The fix described in this paper was built for two new roles — accountant and auditor — that did not exist on the point-of-sale side of the product before this year. Building it required changing the general authorization check that every logged-in user passes through, and that check turned out to affect more than the two new roles. A separate, older set of team roles — analyst, buyer, viewer — invited under the product's existing business-intelligence system, had been passing through that same check, and receiving the same silent substitution described in Section 4, for as long as that older system had existed: an invited team member with one of those roles, visiting the point-of-sale side of the product, would have quietly seen their own empty account rather than an honest refusal.

The fix changes that behaviour for those roles too, and does so openly rather than quietly: a team member in one of those categories now receives a clear access-denied response instead of a misleadingly empty dashboard. This is very likely the correct outcome. It is also a visible behaviour change for anyone who had, for whatever reason, come to rely on the old response, and it is disclosed here as exactly that — a side effect discovered in the course of fixing something else, not a benefit planned from the start.

9. Discussion and Limitations

This paper carries the same caveats as the one before it, for the same reasons.

The fix described here has not, as of this writing, been exercised by a real delegated account in production. It has been verified against the product's own type system and build process, and reasoned through carefully against the code paths it touches, but no accountant or business partner has yet actually been invited, in the ordinary course of a real business's use of the product, to confirm that the experience matches what is described here. That is the single most important open item this paper is reporting rather than resolving.

The competitive comparison in Section 7 is limited to publicly documented behaviour of a small number of products, checked once, at a single point in time; software changes, and a gap observed today is not a permanent one. The claim there is about a pattern, not a permanent verdict on any named product.

Finally, this paper — like the two before it — is a single company's account of its own decisions, evaluated against its own reasoning and its own operational history, without independent replication. It is offered in that spirit: not as proof that this is the correct way

to model delegated trust in small-business software, but as a specific, honestly reported account of one attempt, for others building in similar conditions.

10. Conclusion

The first paper from this desk argued that micro informal enterprises need security before they need formality. The second argued that software meant to serve them has to earn a comparable security in its own interface before any feature inside it can help anyone. This paper has argued a third, adjacent point: once that software is actually in use, trust is not a single grant an owner hands out in one size. An accountant, a business partner, and a hired cashier are three different relationships, built on three different mixes of visibility and control, and software that only offers “owner” or “staff” will eventually force one of them into a box that does not fit — usually, as this paper has shown, by quietly showing them nothing at all, rather than by refusing them honestly.

The most useful evidence for that argument, again, is the least flattering: a role literally called “accountant” existed in this product’s own records for months before it granted access to anything a real accountant would need to see. Building the fix took more time thinking about which of two kinds of trust a given relationship actually called for than it took writing the code that enforces the answer. That ordering is, if this paper has a single recommendation to hand to the next person building software for this population, the one worth keeping: work out what kind of trust a relationship actually is before deciding how much of it to grant.

Access control · Financial trust · Point of sale · Micro-enterprise · East Africa

---