baanjai

ELEVATION, REIMAGINED

LEAD MAGNET · FOR MSPS

The MSP Margin-Leak Playbook

Find the hours admin requests are quietly stealing from every client contract — and plug the leak.

A step-by-step field guide · ~6 pages · measure, standardize, productize, prove

Information is free. Implementation is paid.
A field guide from the team behind Baanjai.

DM MSP on Instagram

> **Why this matters.** You win clients on security, then quietly give the margin back one admin request at a time. "Can you make me an admin?" and "can you install this?" are the most expensive tickets you run — high volume, low value, and they don't scale. This playbook shows you how to measure that leak in dollars, plug it with one repeatable workflow across every tenant, and turn the fix into a security tier clients happily pay for.

What you'll achieve

- A real dollar figure for the admin-request leak, per client and across the book.
- One standardized least-privilege off-ramp that works the same in every tenant.
- A packaged, priced security tier built on it — new MRR, not just saved cost.
- A QBR proof play that shows clients value on a screen and renews contracts.

Phase 1. Measure the leak

You can't charge for — or fix — what you haven't quantified. Start in your PSA.

1 Tag every admin/install ticket for one month

Add a PSA ticket type or tag (e.g. `elevation-request`) and apply it to anything that's really "grant me admin / install this for me / I can't install X." Train the techs to tag at close. One month gives you a defensible baseline.

2 Convert volume to dollars

For each client compute the monthly leak:

```
leak_$ = tickets/mo × avg_handle_time(hrs) × loaded_tech_rate($/hr)
```

Worked example: 60 elevation tickets/mo × 0.25 hr × $90/hr = **$1,350/mo** (~$16k/yr) of labor on one client — before context-switching cost, which is real and larger than the raw minutes.

> **Watch out:** the hidden cost isn't the 15 minutes — it's the interrupt. Every elevation ticket pulls a tech off billable project work; the true drag is 2–3× the handle time. Note it even if you only count the raw hours.

Phase 2. Map it to the contract

Now compare the leak to what you actually priced.

3 Find the margin erosion per client

Lay leak-hours next to the monthly hours the agreement was priced on. Most MSPs find admin requests alone consume **25–50%** of an SMB block — margin you're funding out of pocket. Rank clients by leak: your biggest

leakers are both your best ROI to fix and your best upsell candidates.

Phase 3. Design ONE multi-tenant off-ramp

A per-client snowflake process is why this hurts. Standardize once, deploy everywhere.

4 Define a single approval workflow you run identically in every tenant

1. **Self-service software catalog** for common apps so most requests never reach a tech.
2. **Request-to-elevate** for the rest: scoped to the action, auto-expiring, approved by a tech or a designated client champion — with a target SLA (e.g. < 5 min in hours).
3. **Per-machine rotating break-glass** (Windows LAPS) for the rare true-admin case — never a shared password reused across clients.
4. **One report template** that proves, per tenant, that nobody holds standing local admin.

> **Watch out:** reused or shared local-admin credentials across clients are a lateral-movement disaster and an MSP-wide liability — one popped client becomes a breach across your book. Per-machine, rotating, retrieved-on-demand only.

Phase 4. Productize and price it

Don’t just save the cost — sell the outcome.

5 Package the off-ramp as a security tier

- Name it plainly: "Least-Privilege / No Standing Admin," ideally bundled with hardening + patch reporting into a security add-on.
- Price it as per-seat or per-endpoint MRR. The pitch writes itself: it removes the admin-rights risk class *and* the ticket noise, and it’s provable on a screen.
- Position it as table stakes for cyber-insurance and compliance — you’re helping them answer the questionnaire, not selling them software.

> **Watch out:** don’t bury this in the base plan for free. The clients who need it most are the ones who’ll happily pay for "we made the auditor and the insurer go away."

Phase 5. Roll out across the book

Sequence it so wins compound.

6 Deploy biggest-leaker first, templatize, repeat

- Start with your highest-leak client as the reference rollout; capture the before/after numbers.

- Turn that into a repeatable onboarding runbook (same workflow, same report) so each subsequent tenant is hours, not days.
- Use the reclaimed tech hours to absorb the rollout — the project largely pays for itself in recovered capacity.

Phase 6. The QBR proof play

Stop telling clients you keep them safe. Show them.

7 Run the same 3-slide play in every review

1. **Risk removed:** the screen showing zero (or fully-justified) standing local admins on their fleet.
2. **Noise removed:** elevation handled via fast self-service/approval — fewer interruptions for their staff.
3. **Proof on demand:** the report they can hand to their insurer/auditor.

That screen is the single most renewal-protecting artifact you own. It reframes you from "the people who fix things" to "the people who measurably reduced our risk."

KPIs to track

- Elevation tickets per endpoint per month (target: down 70%+).
- Standing local admins per client (target: ~0 + justified exceptions).
- Security-tier attach rate and added MRR.
- Reclaimed billable hours per tech per month.

Common objections (and answers)

CLIENT SAYS	YOU SAY
"My people need to install their own software."	"They still can — from a vetted catalog, instantly. The difference is malware can’t, and a phished account can’t take the whole company down."
"This sounds like more friction."	"Approvals land in seconds and most installs are self-service. The friction you feel today is the ticket queue; this shrinks it."
"Why does this cost extra?"	"It removes an entire risk class and answers your insurer’s hardest questions. It’s the control that lowers your premium and your breach odds."

How Baanjai changes this

THE WORKFLOW YOU STANDARDIZE ON — ALREADY BUILT

Phase 3 asks you to design one approval workflow, a LAPS break-glass, and a per-tenant proof report, then run them identically everywhere. That's precisely what Baanjai is — multi-tenant by design, so the margin-leak fix is also the product you bill for.

DOING IT BY HAND	WITH BAANJAI
Tag tickets in the PSA and hand-calculate the leak per client.	Elevation activity is logged per tenant — the volume (and the savings story) is reportable, not a spreadsheet exercise.
Stand up a different approval hack in each client and pray the SLA holds.	One request-to-elevate workflow across every tenant: scoped, auto-expiring, approved from your phone in seconds.
Manage shared or LAPS break-glass passwords across dozens of clients.	On-demand time-boxed elevation removes most break-glass; what remains is per-machine and centrally controlled.
Assemble a custom "you're safe" deck for each QBR.	A live per-client proof screen (no standing admin + full approval trail) you can show in any review.
Saved cost, but nothing new to sell.	A packaged, provable security tier = new recurring revenue, not just recovered hours.

BOTTOM LINE

The playbook turns a hidden cost into a managed, billable service. **Baanjai is the engine that makes it identical across every client** — so you stop bleeding margin on admin grants and start charging for the screen that proves it.

What Baanjai is

Baanjai removes standing local admin and ransomware risk from one pane of glass: request-to-approve elevation tnat's scoped and auto-expiring, endpoint hardening (Controlled Folder Access, shadow-copy & LSA protection, application control), patch policies, and a live "who-has-what" view you can hand to an auditor. Everything in this playbook works by hand — Baanjai makes it a Tuesday.

Run the numbers with us → baanjai.app/contact