baanjai

ELEVATION, REIMAGINED

LEAD MAGNET · INTERNAL IT

The Local-Admin Offboarding Playbook

Remove standing local admin across your fleet in 30 days — without a helpdesk revolt.

A step-by-step field guide · ~7 pages · tooling, commands, rollout & proof

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

DM ADMIN on Instagram

> **Why this matters.** Standing local-admin rights are the single highest-leverage risk on your fleet. A phished user with admin can install anything, disable defenses, and delete your shadow copies; the blast radius of any one account equals the rights you left on it. This playbook takes you from "we think most people are admins" to "here is the screen that proves who is, and why" — in about 30 days, without a helpdesk revolt.

What you'll achieve

- A complete, exported inventory of every local administrator across the fleet.
- A fast, low-friction approval path so removing admin doesn't create a wall of tickets.
- Enforcement that keeps admin off (and detects re-additions) instead of a one-time cleanup that drifts back.
- An audit- and insurer-ready report you can produce on demand.

Before you start — prerequisites

- An account that can query and modify membership of the local `Administrators` group fleet-wide (Domain Admin / Intune admin, or your RMM running as SYSTEM).
- A way to push policy: Group Policy (on-prem AD), Microsoft Intune (cloud/hybrid), or an RMM.
- A break-glass plan for true local-admin needs (covered in Phase 3) — ideally Windows LAPS, the built-in successor to legacy Microsoft LAPS (GA April 2023).
- Buy-in from one cooperative team to pilot. Don't start with the executives.

Phase 1. Discover — get the real list

You cannot remove what you cannot see. AD group names lie; query the machines directly.

1 Enumerate local admins on one machine

The built-in cmdlet lists every member of the local Administrators group, including nested AD/Entra groups:

```
Get-LocalGroupMember -Group "Administrators"
```

The `PrincipalSource` column (Windows 10 / Server 2016+) tells you whether each entry is Local, ActiveDirectory, or AzureAD — note it, you'll classify on it.

2 Sweep the whole fleet

Run it remotely and collect to one CSV. For a handful of machines:

```
Invoke-Command -ComputerName PC01,PC02 -ScriptBlock {
  Get-LocalGroupMember -Group 'Administrators' |
    Select-Object @{n='Computer';e={$env:COMPUTERNAME}}, Name, PrincipalSource, ObjectClass
}
```

For the full estate, drive it from your AD computer list and export:

```
$pcs = (Get-ADComputer -Filter 'Enabled -eq $true').Name
Invoke-Command -ComputerName $pcs -ScriptBlock {
  Get-LocalGroupMember -Group 'Administrators'
} | Export-Csv .\local-admins.csv -NoTypeInformation
```


> **Watch out:** machines that are offline, blocked by the firewall, or unreachable by WinRM silently drop out of the results — a missing row reads as "no admins," which is the opposite of true. Track which hosts answered and re-sweep the rest. If you have an RMM, run the same command as a fleet job (as SYSTEM) for fuller coverage.

3 Build the master inventory

Consolidate into one sheet with: computer, principal, source (Local/AD/Entra), type (user/group), and three empty columns you'll fill next — `Bucket`, `Owner`, and `Decision`. This sheet is your project tracker and your first piece of audit evidence.

Phase 2. Classify — sort by *why*

Most of the list isn't "people who need admin." It's drift. Sort it so the easy wins are obvious.

4 Put every grant in one of three buckets

- **(a) Needs admin daily** for a genuine job function (developers compiling drivers, a few IT staff). Smallest bucket.
- **(b) Needed it once** — a one-time install or vendor setup, never revoked. Large bucket, low risk to remove.
- **(c) Nobody knows** why this principal is there. Largest bucket. Treat as removable until proven otherwise.

For each row, also record the **real owner** (a person who can answer "do you still need this?") and a tentative **Decision**: Keep / Remove / Convert-to-on-demand.

> **Watch out:** nested groups. A single "Domain Users"-style group sitting in local Administrators can make your whole company admin in one line. Flag nested groups first — they're your biggest single fixes.

Phase 3. Build the off-ramp *before* you cut

This is the step that decides whether the project succeeds. Removal fails when a user hits a wall and there's no fast, legitimate way over it.

5 Define how a legitimate need gets met in seconds

Decide, and write down, the answer to "I removed your admin — here's what you do when you genuinely need it":

1. **Self-service install catalog** for the common asks (Company Portal / your RMM software center) so 80% of requests never need elevation at all.
2. **A request-to-elevate path** for the rest: the user requests, an approver (you, or a team lead) approves, and the elevation is **scoped to that action** and **auto-expires**. Define the target approval SLA (e.g. < 5 minutes during business hours) — if approval is slow, people will route around you and the project dies.
3. **Break-glass** for true local admin (a tech on-site, an offline machine): a per-machine, rotating local admin password via **Windows LAPS**, retrieved only when needed and rotated after use:


```
Get-LapsADPassword -Identity PC01 -AsPlainText
```

> **Watch out:** a shared, never-changing local admin password is how one compromised machine becomes all of them (pass-the-hash). LAPS gives every machine a unique, rotating password — set it up before you remove rights, not after.

Phase 4. Choose your enforcement mechanism

A one-time manual cleanup drifts back within weeks. Pick a mechanism that keeps admin off and re-applies on every refresh.

6 Pick the tool that fits your environment

MECHANISM	BEHAVIOUR & WHEN TO USE
Group Policy Preferences → Local Users and Groups	**Recommended for mixed fleets.** Selectively *removes* specific principals and *adds* approved ones without wiping the whole group. Path: `Computer Configuration → Preferences → Control Panel Settings → Local Users and Groups`.
Restricted Groups	Authoritatively *overwrites* the entire Administrators membership to exactly what the GPO says and resets it on every refresh. Powerful but blunt — only where every machine should have identical membership. Path: `Computer Configuration → Policies → Windows Settings → Security Settings → Restricted Groups`.
Microsoft Intune	For cloud/hybrid: `Endpoint security → Account protection → Local user group membership` to manage the Administrators group, paired with Windows LAPS policy.

Manual removal for reference (one principal, one machine):

```
net localgroup administrators "DOMAIN\jdoe" /delete
```

> **Watch out:** never empty the local Administrators group or remove the built-in Administrator + Domain Admins. Always leave a known-good admin path (and your LAPS break-glass) or you will lock yourself out of the machine.

Phase 5. Pilot with the friendliest team

Prove the off-ramp on people who will tell you the truth, not the people who will escalate to your boss.

7 Remove admin from 5–10 cooperative users and measure

Scope the policy to a pilot group and watch four numbers for two weeks:

- Install/elevation requests per user per week.
- Median time-to-approve.
- How many requests escalated or were worked around.
- Tickets that turned out to need *no* elevation (catalog gaps).

Tune the catalog and the approval SLA until friction is near zero. A simple comms line that works: *"We're removing standing admin so that if your account is ever phished, the damage stops at you. Need to install something? Here's the 30-second path."*

Phase 6. Roll out in waves

Department by department, communicating the win each time.

8 Expand by group, keep a rollback switch

- Sequence waves from lowest- to highest-friction departments; leave executives and developers (bucket a) for last, with tailored on-demand rules.
- Publish the standing-admin count dropping after each wave — it's your proof and your momentum.
- Keep the policy scoped so you can pause a wave instantly if something breaks.

Phase 7. Prove it — on demand

The finish line isn't zero admins. It's being able to answer the question in one screen.

9 Stand up a repeatable report

Re-run your Phase 1 sweep on a schedule and diff it against the approved baseline. Any new local admin that appears between runs is either an approved exception (with an owner and a reason) or an incident to investigate. That recurring diff — plus your request/approval log — is exactly what answers *"who has admin, and why?"* for an auditor or a cyber-insurance questionnaire.

> **Watch out:** a point-in-time screenshot ages instantly. Auditors and insurers increasingly want to see that the control is *continuous* — a live view beats a quarterly export.

The 30-day cadence

DAYS	FOCUS
1–5	Discover + build the master inventory (Phases 1–2).
6–12	Stand up the off-ramp: catalog, approval path + SLA, Windows LAPS break-glass (Phase 3).
13–18	Configure enforcement + pilot one team; tune to near-zero friction (Phases 4–5).
19–28	Roll out in waves, publish the dropping admin count (Phase 6).
29–30	Stand up the recurring proof report (Phase 7).

Common pitfalls

- **Cutting before the off-ramp exists.** Guarantees a ticket flood and a reversal. Phase 3 first, always.
- **Drift.** A manual cleanup with no enforcement is admin-creep waiting to happen. Use a policy that re-applies.
- **The angry-VP rollback.** One loud exec can undo months. Have a fast, dignified on-demand path ready for them specifically.
- **Shared local-admin passwords.** Replace with LAPS before you remove anything.

How Baanjai changes this

FROM A 30-DAY PROJECT TO A SETTING

Every phase above is real work with scripts, GPOs, a LAPS rollout, and a reporting pipeline you maintain. Baanjai collapses the hard parts — the off-ramp, the enforcement, and the proof — into one place, so least privilege is something you *operate*, not a project you survive.

DOING IT BY HAND	WITH BAANJAI
Sweep the fleet with scripts; offline/WinRM-blocked machines silently drop out.	Every endpoint reports its admins through the agent — one live, complete inventory, no missed hosts.
Build an approval path in a ticketing tool; hope the SLA holds so people don't route around you.	Built-in request-to-elevate: the user asks, you approve from your phone, and the elevation is scoped to that action and auto-expires.
Stand up Windows LAPS for break-glass and rotate passwords manually.	On-demand, time-boxed elevation replaces most break-glass; no standing local admin to manage at all.
Maintain GPOs / Intune policies and re-run diffs to catch drift.	Policy is enforced continuously and re-additions surface automatically — drift can't accumulate.
Produce a point-in-time screenshot at audit time and hope it's current.	A live "who-has-admin, and why" view with a full approval trail — answer the audit question in one screen, any day.

BOTTOM LINE

The playbook gets you to least privilege. **Baanjai keeps you there** — and turns "who has admin, and why?" from a paragraph that starts with "well…" into a screen you can show on a Tuesday.

What Baanjai is

Baanjai removes standing local admin and ransomware risk from one pane of glass: request-to-approve elevation that'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.

See it on your fleet → baanjai.app/contact