SAMPLE REPORT EXCERPT | Findings below are fictionalized and anonymized for illustration. Structure, depth, and rating method match a real deliverable.

Salesforce Architecture Assessment

Sample Report Excerpt | Aetrum LLC | aetrumllc.com

About this excerpt

The full assessment reviews an org across six frameworks: data architecture; sharing and security; identity and access; automation and code; integrations and deployment; monitoring and governance. The full report contains a complete findings register, a target-state architecture, a roadmap, and a one-page executive summary. This excerpt shows the executive summary format, four sample findings, and a roadmap fragment. The sample org is a fictional company with roughly 120 Salesforce users, six years in production, and five external integrations.

Executive summary (sample)

The org supports the business today, but it carries risk in three places. First, access is broader than the business intends: sensitive records are visible to most users because org-wide defaults were never tightened after go-live. Second, one integration account holds administrator-level rights and authenticates with a password instead of OAuth; a leak would expose data org-wide and an expiry would halt three integrations at once. Third, automation on the busiest object has grown in three overlapping layers, which makes every change slower and riskier than it should be. None of this requires a rebuild. The roadmap closes the highest risk within thirty days and fully remediates the second within ninety, with no user disruption.

How findings are rated

Risk	Business impact if unaddressed: High, Medium, Low
Effort	Estimated remediation effort: Small (days), Medium (weeks), Large (a quarter or more)
Evidence	Every finding cites what was inspected and how to reproduce it, so your own team can verify or challenge it

Sample findings

Finding 1: Integration user holds admin-level rights with password authentication — **High risk** | Small effort

Framework: Identity and access

What we observed: One integration account is assigned Modify All Data and authenticates with a username and password stored in the middleware. Login history shows this account is used by three separate external systems.

Why it matters: Because the account holds Modify All Data, a leaked credential exposes every record in the org. An expired or rotated password takes down all three integrations at once. Shared credentials also leave no reliable way to attribute a given change to a specific system.

Recommended fix: Replace with one External Client App per system using the OAuth 2.0 client credentials flow, which issues no refresh tokens, so a stolen token cannot be replayed. Salesforce now recommends External Client Apps and, as of Spring '26, no longer allows new connected apps to be created unless Salesforce Support grants an exception; the legacy username-password flow is not used. Run each as a dedicated integration user on an API-only Integration User license with a minimum-access permission set. Sequence: create, test in sandbox, cut over one integration per week.

Finding 2: Org-wide defaults leave sensitive records readable by all users — **High risk** | Medium effort

Framework: Sharing and security
What we observed: Org-wide default for two objects holding sensitive customer data is Public Read/Write, and the profiles in use grant broad object access to both. The two gates that could restrict visibility are both open, so access is effectively flat across all 120 users, including seven third-party contractor logins.

Why it matters: Any user whose profile grants these objects, which here is all of them, can read and edit records the business considers restricted. This is the finding a customer security review or auditor will flag first.

Recommended fix: Move the two objects to Private, then grant access back deliberately through role hierarchy and criteria-based sharing rules. Test in a full sandbox with a permissions comparison report before and after, so no legitimate access breaks.

Finding 3: Three overlapping automation layers on the busiest object — **Medium risk** | Medium effort

Framework: Automation and code
What we observed: The Case object carries two active Apex triggers, four record-triggered Flows, and one Workflow Rule left over from a retired automation layer. Two of the Flows and one trigger update the same field under different conditions. There is no trigger framework, no automation inventory, and no defined trigger ordering across the layers.

Why it matters: Every new automation risks colliding with an existing one. The team reports that small changes take days to test, and two production incidents in the last year trace back to automation firing in an unexpected order.

Recommended fix: Consolidate to one trigger per object behind a simple trigger framework, migrate the Workflow Rule into an existing Flow, and document one entry point per operation. This is refactoring, not new function, so it can proceed in slices without user-facing change.

Finding 4: No usable record of what depends on which credential or certificate — **Medium risk** | Small effort

Framework: Monitoring and governance
What we observed: Certificates, named credentials, and connected apps are undocumented. Two certificates expire within ninety days; nobody interviewed could say which integrations they serve without investigating.

Why it matters: Salesforce emails admins before a certificate expires, but the alert names only the certificate; nothing maps it to the integrations that depend on it or to an owner. The next rotation surfaces as an outage with no clear blast radius instead of a planned change.

Recommended fix: Build a dependency register mapping each credential and certificate to the components and integrations that use it, with owners and expiry dates, then add expiry reminders at ninety, sixty, and thirty days. Building the register for an org this size is about a day; keeping it current is minutes per change.

Roadmap fragment (sample)

This month	Rotate the shared integration credential to OAuth (Finding 1). Start the org-wide default change in sandbox (Finding 2). Build the dependency register (Finding 4).
Next 90 days	Complete the sharing model change with before/after verification. Consolidate Case automation into one framework (Finding 3).
Beyond	Target-state items from the full report: deployment pipeline, monitoring baseline, and data model cleanup, sequenced against the team's release calendar.