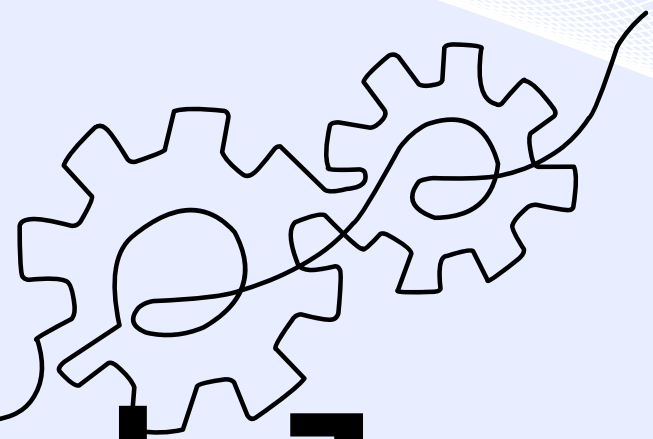

Tech Jargon *Part 3*

for

Product Managers

Who Gets In & What They See

Auth | SSO | OAuth | RBAC | Tokens

From an Engineer-Turned-PM

The Security & Data Isolation Mental Model

A way of understanding your product through who gets in, what they can do, and how safely each customer's data stays separated inside a shared system.

It helps PMs design authentication, permissions, and tenant boundaries as product choices—not IT checkboxes. Get this wrong: compliance failures, Get it right: competitive advantage.

When you understand this mental model, you can:

- Ask precise questions about login, roles, and approvals instead of saying "just give them access."
- Design flows that respect tenant boundaries and least-privilege access, preventing data leaks between customers.
- Work with engineers on permission models, SSO/MFA, and audit logs that pass enterprise security reviews without blocking usability.

This guide covers the 10 essential terms that unlock this way of thinking.

Authentication

Verifying a user's identity before letting them in

Authentication is how the system checks that a user is who they claim to be—typically with email/password, SSO, or MFA.

Think of it as checking someone's ID at the front door: you don't care why they're visiting yet, just whether they are who they say they are.

Authentication is working, but we still need MFA before we roll this out to admins.

For PMs, this means you need to be clear on how different users sign in, how strong security must be for each group, and what happens when they forget credentials or change devices.

SSO

Single sign-on

Single sign-on lets users log in with an identity provider like Okta, Azure AD, or Google Workspace and reuse that login across multiple apps.

Think of it as one company badge that works across multiple office buildings instead of juggling separate keys for each one.

The customer needs SSO via Okta before they'll roll us out company-wide.

For PMs, this means SSO can be a deal-breaker for enterprise customers, so you must know which providers you support, how it affects onboarding, and where it sits on your roadmap

MFA

Multi-factor authentication

MFA requires two or more factors to log in—something you know (password), something you have (code, app, hardware key), or something you are (biometrics).

Think of it as needing both a key and a PIN to open a safe—stealing just one isn't enough to get in.

Finance users must have MFA enabled before they can access billing data.

For PMs, this means you balance friction and security: decide which roles require MFA, how to roll it out without tanking adoption, and how recovery flows work when people lose access.

Authorization

Controlling what an authenticated user can do

Authorization defines what a signed-in user is allowed to see and do—what data they can access and which actions they can take.

Think of it as deciding which rooms a badge opens once someone is inside the building: lobby access is not the same as access to the vault.

Auth works, but our authorization rules are too broad—view-only users can still edit reports.

For PMs, this means you design roles and permissions up front, so you don't bolt on restrictions later after a security incident or angry customer.

RBAC

Role-based access control

RBAC assigns permissions to roles (Admin, Manager, Viewer) and then assigns users to those roles.

Think of it as issuing standard “manager badges” and “visitor badges” instead of hand-crafting a unique badge for every single person.

Let's define clear RBAC roles so customer admins can manage access themselves.

For PMs, this means a good role model simplifies your UX, reduces support tickets about access, and makes you easier to buy in complex organizations.

Permission model

The overall structure of roles, scopes, and rules that govern access

The permission model is the big picture of how access is decided—roles, scopes, resources, and any exceptions.

Think of it as the building's access policy: who can enter via which door, at what times, and with what level of clearance.

We need to rethink our permission model; it doesn't match how customers' orgs actually work.

For PMs, this means you co-design this with engineering and customers early, because it shapes navigation, workflows, and what's even feasible to build later.

Least privilege

Giving users the minimum access they need to do their job

Least privilege means users get the minimum permissions needed to do their job—no more.

Think of it as giving someone a key to their own office, not a master key to every door in the building "just in case."

Support doesn't need full admin rights—let's scope them to least-privilege roles.

For PMs, this means you design defaults that are safe by design: avoid everyone-is-admin setups, and be ready to explain to customers why tighter permissions protect them.

Multi-tenant SaaS

One app, many customers, separated logically

In a multi-tenant SaaS, multiple customer organizations share the same application and infrastructure, but their data and configuration are kept logically separate.

Think of one apartment building where many families live in different units, sharing elevators and plumbing but not living rooms or mailboxes.

We're a multi-tenant SaaS, so all customers use the same app, but their data must stay isolated.

For PMs, this means every feature—search, reporting, AI—must respect tenant boundaries so no customer ever sees another customer's data.

Tenant isolation

Preventing data leaks between customers in a shared system

An audit log records important actions—who did what, to which resource, and when.

Think of it as CCTV plus a sign-in sheet for your app: a timeline of who changed what, where, and when.

Before we sell into this regulated customer, we need stronger tenant isolation guarantees.

For PMs, this means you treat isolation requirements as first-class: they influence architecture, certifications, and how you answer security questionnaires.

Audit log

Record of who did what and when, for security and compliance

An API key or token is a credential a client includes with requests so the API can identify and authenticate who's calling. It's like a password for systems, often tied to rate limits and permissions.In practice, API keys are often long-lived, while tokens are usually short-lived and scoped to specific permissions.

Think of it as a backstage pass: it proves you're authorized to access certain areas and determines what you're allowed to do.

Security needs an audit log so they can see who changed permissions and when.

For PMs, this means audit logs are not just "nice for compliance"; they're a feature you can sell to security-conscious customers and a tool for debugging sensitive issues.

Stay Tuned

Up Next

A plain-English guide to understanding how past shortcuts shape today's options—and why "just refactor it" isn't always simple.