Engineer

RBAC

Berechtigungen für Rollen statt für einzelne Personen

Was es ist

RBAC entscheidet anhand der zugewiesenen Rollen, was jemand tun darf, statt anhand seiner Identität. Die Buchstaben stehen für role-based access control, rollenbasierte Zugriffskontrolle. Das NIST, bei dem das Modell formalisiert wurde, nennt es auch „rollenbasierte Sicherheit“. Eine Rolle ist eine benannte Sammlung von Berechtigungen, etwa „Sachbearbeitung in der Buchhaltung“ oder „nur lesender Zugriff auf Produktionssysteme“. Benutzern werden Rollen zugewiesen, und jede von einem Benutzer ausgeübte Berechtigung stammt aus einer davon.

David Ferraiolo und Rick Kuhn vom NIST schlugen das allgemeine Modell 1992 als Alternative zur damals üblichen obligatorischen und diskretionären Zugriffskontrolle vor. 1996 teilten Sandhu, Coyne, Feinstein und Youman es in Schichten auf: RBAC0 mit Benutzern, Rollen und Berechtigungen, RBAC1 zusätzlich mit Rollenhierarchien, RBAC2 zusätzlich mit Einschränkungen und RBAC3 mit beidem. Die beiden Entwicklungslinien wurden vereint und gingen in den Standard ANSI/INCITS 359-2004 ein, der 2012 überarbeitet wurde. Sein Referenzmodell hat vier Bestandteile. Core RBAC, der einzige für jedes System vorgeschriebene Bestandteil, enthält die Zuordnungen von Benutzern zu Rollen und von Berechtigungen zu Rollen sowie die Sitzung, in der ein Benutzer einen Teil seiner zugewiesenen Rollen aktiviert. Hierarchisches RBAC lässt eine übergeordnete Rolle eine untergeordnete erben. Statische Funktionstrennung beschränkt, welche Rollen einem Benutzer gemeinsam zugewiesen werden dürfen. Dynamische Funktionstrennung beschränkt, welche Rollen in einer Sitzung gleichzeitig aktiv sein dürfen. So kann jemand sowohl die Rolle zum Anlegen einer Zahlung als auch die zu ihrer Freigabe innehaben, ohne beide in derselben Sitzung zu verwenden.

Der Unterschied zu einer gewöhnlichen Gruppe ist kleiner, als er aussieht. Die NIST-FAQ erklärt ihn so: Eine Gruppe ist üblicherweise eine Sammlung von Benutzern, und die meisten Gruppenmechanismen erlauben auch, eine Berechtigung direkt an einen Benutzer zu hängen. Die FAQ nennt dies ein „Schlupfloch“, das die Beziehung zwischen Benutzern und Berechtigungen schwer kontrollierbar macht. RBAC schließt es, indem Berechtigungen ausschließlich an Rollen hängen. Den Grund nennt eine weitere Zeile derselben FAQ: Rollen in einer Organisation bleiben recht stabil, während Benutzer und Berechtigungen zahlreich sind und sich schnell ändern. Alle Zugriffe über Rollen zu führen vereinfacht daher Verwaltung und Überprüfung.

Datenbanken setzen das Modell oft direkt um. PostgreSQL kennt seit Version 8.1 nur eine Art von Entität, die Rolle. Sie kann als Benutzer, als Gruppe oder als beides dienen. Eine als Gruppe gedachte Rolle hat üblicherweise kein LOGIN; die Mitgliedschaft wird wie jedes andere Privileg gewährt. Zwei Mitgliedschaftsoptionen steuern, wie ein Mitglied die Privilegien der Gruppe nutzt: INHERIT gibt sie ihm automatisch, und SET erlaubt den Wechsel in die Rolle mit SET ROLE. Das kommt der RBAC-Idee nahe, eine Rolle nur in der Sitzung zu aktivieren, die sie benötigt.

```
-- the role carries the permissions; nobody logs in as it
CREATE ROLE reporting NOLOGIN;
GRANT USAGE ON SCHEMA sales TO reporting;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO reporting;

-- a person receives the role, never the table
CREATE ROLE alex LOGIN;
GRANT reporting TO alex;

-- a powerful role that has to be switched on deliberately
CREATE ROLE schema_admin NOLOGIN;
GRANT schema_admin TO alex WITH INHERIT FALSE;
-- later, in the one session that needs it: SET ROLE schema_admin;
```


Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist RBAC?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Konten, Passwörter und Berechtigungen werden so geordnet, dass alle genau den beruflich erforderlichen Zugriff haben und ihn am Tag ihres Ausscheidens verlieren.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. 1.000 RBAC-Regeln für sichere Linux-Server — 1,000 Zugriffsregeln zum Schließen einer Sicherheitslücke

1.000 RBAC-Regeln für sichere Linux-Server Kontakt aufnehmen →

Warum es wichtig ist

Zugriffskontrolle muss zwei Fragen beantworten, die in verschiedene Richtungen weisen: Ist diese Anfrage jetzt erlaubt, und wer darf im gesamten System was tun? Die meisten Ansätze beantworten die erste gut. RBAC ist für die zweite gebaut, denn seine Antwort ist eine kurze Liste von Rollen und ihren Mitgliedern, die ein Auditor oder eine Führungskraft tatsächlich lesen kann. Das NIST fasst die Alternative, attributbasierte Zugriffskontrolle, bei der jede Entscheidung Regeln über Attribute des Benutzers, der Ressource und des Kontexts auswertet, in einem Satz zusammen: Sie „kann leicht einzurichten, aber komplex zu verwalten sein“.

Die praktische Entscheidung ist, wo Rollen nicht mehr ausreichen. Das Modell von 1992 gewährt Zugriff „nur wenn“ die Rollenanforderungen erfüllt sind. Diese Formulierung lässt bewusst Raum für weitere Prüfungen beim Zugriff. Eine Arztrolle erlaubt das Lesen von Patientenakten, aber nicht der Akte jedes Patienten: Die Beziehung zwischen diesem Arzt und diesem Patienten ist eine Eigenschaft der Daten, keine Berufsbezeichnung. Dort gehört ein zweiter Mechanismus hin, etwa zeilenbezogene Sicherheit in der Datenbank oder eine Attributregel in der Anwendung, als Ergänzung der Rollen statt als deren Ersatz.

Die Empfehlung lautet, Zugriff zuerst über Rollen zu modellieren, deren Namen tatsächliche Tätigkeiten beschreiben. Berechtigungen hängen nur an Rollen und niemals direkt an einem Anmeldekonto. Halten Sie die Rollenliste so kurz, dass eine Person sie vollständig lesen kann. Ergänzen Sie zeilenbezogene oder attributbasierte Regeln nur dort, wo eine Rolle die Grenze nachweislich nicht ziehen kann. Lassen Sie die Datenbank alles durchsetzen, was sie durchsetzen kann: Eine Datenbankprüfung gilt für jeden Client, einschließlich des vergessenen Codepfads.

Wie es in der Praxis aussieht

Bei einer Tätigkeit im Datenbankengineering von 2022 bis 2024 wurde der Zugriff auf jede Entwicklungsumgebung, Anwendungsinstanz, PostgreSQL- und MS-SQL-Datenbank sowie jeden Linux-Server durch 1.000 RBAC-Regeln geregelt. Das Modell war der einfache Teil; die Menge war die Herausforderung. Die

Berechtigungen wurden in wiederverwendbare Kategorien wie „nur lesender Zugriff auf Produktionsdatenbanken“ zerlegt und Rollen wie Entwicklern, Anwendungsinstanzen und Datenbankadministratoren zugeordnet. Entwickler erreichten nur ihre vorgesehenen Server. Anwendungsinstanzen erhielten begrenzte Berechtigungen, um laterale Bewegungen zu verhindern. Ansible setzte die Regeln in allen Umgebungen einheitlich um. Die Regeln wurden dokumentiert, damit Teams sie prüfen oder ändern konnten. Geliefert wurden der rollenbasierte Regelsatz und ein wiederholbarer Weg, ihn anzuwenden und zu überprüfen.

Rollen wirken selten allein. Als eine Kartenanwendung von MSSQL auf PostgreSQL mit PostGIS wechselte, wurden rollenbasierte Zugriffskontrollen zusammen mit zeilenbezogener Sicherheit eingeführt, um Datenschutzrichtlinien über mehrere Benutzergruppen hinweg durchzusetzen. Diese Kombination entspricht der üblichen Arbeitsteilung: Rollen bestimmen, auf welche Tabellen eine Gruppe zugreifen darf; Zeilenrichtlinien bestimmen, auf welche Zeilen darin.

Die strengste Variante entzieht den Zugriff auf die Tabellen vollständig. In einer funktionsbasierten PostgreSQL-Datenschicht aus einer seit 2025 laufenden Tätigkeit ist jede Anwendungsoperation eine von 1.275 Funktionen in 34 Schemas. Die Rolle, mit der sich die API anmeldet, mariner, besitzt EXECUTE auf den Anwendungsfunktionen und USAGE auf den Schemas und sonst nichts: ungefähr 4.000 explizite Berechtigungszuweisungen statt einer pauschalen. Die Funktionen laufen als SECURITY DEFINER, gehören einer getrennten Rolle ohne Anmelderecht und haben einen fest vorgegebenen search_path. Das Superuserkonto bleibt Migrationen und cron vorbehalten. Auch eine kompromittierte API wäre auf das beschränkt, was die Funktionen erlauben. Der Preis ist ebenfalls festgehalten: Ein neues Feld erfordert eine Migration und eine Funktionsänderung.

Wo es schiefgeht

- **Rollen vermehren sich, bis niemand sie mehr überblickt.** Jede Ausnahme wird eine neue Rolle, etwa „Entwickler außer Abrechnung“ oder „Entwickler plus ein Bericht“. Bald gibt es mehr Rollen als Menschen. Diese Rollenexplosion verfehlt den Grund für RBAC: eine zum Überprüfen ausreichend kurze Liste. Geben Sie einer Person zwei oder drei Rollen, statt für jede Ausnahme eine neue zu schaffen. Führen Sie Rollen zusammen, deren Berechtigungsmengen identisch geworden sind.

- **Eine Berechtigung landet direkt auf einem Anmeldekonto.** Ein dringendes Ticket, ein GRANT direkt an einen Benutzer, und dieser hat Zugriff, den keine Rollenprüfung zeigt. Das ist das vom NIST beschriebene Schlupfloch in Gruppensystemen, von Hand nachgebaut. Gewähren Sie Berechtigungen nur Rollen und lassen Sie die Prüfabfrage nach Privilegien von Rollen mit Anmelderecht suchen.
- **Voreinstellungen gewähren Zugriff vor jeder Rolle.** In PostgreSQL steht PUBLIC für alle Rollen. Standardmäßig erhält PUBLIC EXECUTE auf neuen Funktionen sowie CONNECT und TEMPORARY auf Datenbanken. Wurde eine Datenbank von PostgreSQL 14 oder älter aktualisiert, kann PUBLIC auch Objekte im Schema namens public erzeugen. Entziehen Sie diese Standardprivilegien bewusst, möglichst innerhalb der Transaktion, die das Objekt anlegt.
- **Rollenzuweisungen wachsen nur an.** Menschen wechseln die Tätigkeit und behalten die alte Rolle neben der neuen. Vorübergehender Zugriff wird dauerhaft, weil niemand für das Entfernen zuständig ist. Jede Zuweisung braucht einen Grund und einen Prüftermin. Mit dem Verlassen einer Tätigkeit sollten die dazugehörigen Rollen entfallen.
- **Manche Konten umgehen die Regeln.** Superuser und Rollen mit BYPASSRLS ignorieren zeilenbezogene Sicherheit. Der Eigentümer einer Tabelle tut dies normalerweise ebenfalls, sofern die Tabelle nicht auf FORCE ROW LEVEL SECURITY gesetzt ist. Verbindet sich eine Anwendung als Eigentümer oder Superuser, wird jede Zeilenrichtlinie zur Dekoration. Verbinden Sie sie deshalb als Rolle, die nichts besitzt.

Weiterführende Lektüre

- NIST-Projekt zu rollenbasierter Zugriffskontrolle — die archivierte Projektseite: Geschichte des Modells, der Standard von 2004 und seine Überarbeitung von 2012.
- NIST-FAQ zu RBAC — Rollen und Gruppen, RBAC0 bis RBAC3, Funktionstrennung und die vier Bestandteile des Standards.
- PostgreSQL: Rollenmitgliedschaft — Gruppenrollen, Gewähren und Entziehen der Mitgliedschaft sowie die Optionen INHERIT und SET.
- PostgreSQL: Zeilensicherheitsrichtlinien — standardmäßige Verweigerung, Rollen, die Zeilenrichtlinien umgehen, und FORCE ROW LEVEL SECURITY.
- PostgreSQL: Privilegien — was PUBLIC standardmäßig erhält und wie es entzogen wird.
- SQL Server: Datenbankrollen — feste und benutzerdefinierte Rollen in einer SQL-Server-Datenbank.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist RBAC?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/rbac/