Engineer

RBAC

rettigheder givet til en rolle, ikke en person

Hvad det er

RBAC afgør, hvad nogen må, ud fra de roller, vedkommende har, og ikke ud fra hvem vedkommende er. Bogstaverne står for role-based access control, på dansk rollebaseret adgangskontrol, og NIST, hvor modellen blev formaliseret, kalder det også »role based security«. En rolle er en navngiven samling rettigheder, for eksempel »bogholder« eller »læseadgang til produktion«. Brugere tildeles roller, og hver rettighed, en bruger udøver, kommer gennem en af dem.

David Ferraiolo og Rick Kuhn fra NIST foreslog den generelle model i 1992 som et alternativ til datidens obligatoriske og skønsbaserede adgangskontrol. I 1996 delte Sandhu, Coyne, Feinstein og Youman den op i lag: RBAC0 med brugere, roller og rettigheder, RBAC1 med rollehierarkier, RBAC2 med begrænsninger og RBAC3 med begge dele. De to spor blev samlet og blev til standarden ANSI/INCITS 359-2004, der blev revideret i 2012. Dens referencemodel har fire komponenter. Core RBAC, den eneste, ethvert system skal have, rummer tildelingen af brugere til roller og af rettigheder til roller samt sessionen, hvor en bruger aktiverer en delmængde af sine tildelte roller. Hierarkisk RBAC lader en overordnet rolle arve en underordnet. Statisk funktionsadskillelse begrænser, hvilke roller én bruger må tildeles samtidig, og dynamisk funktionsadskillelse begrænser, hvilke roller der må være aktive på samme tid i en session, så en medarbejder kan have både rollen, der opretter en betaling, og rollen, der godkender den, uden at bruge begge i samme omgang.

Forskellen fra en almindelig gruppe er mindre, end den ser ud. NIST's FAQ forklarer det sådan: en gruppe er som regel en samling brugere, og de fleste gruppemekanismer lader også en rettighed hænge direkte på en bruger, hvilket FAQ'en kalder et smuthul (»loophole«), der gør forholdet mellem brugere og rettigheder svært at styre. RBAC lukker det ved kun at knytte rettigheder til roller. Begrundelsen står et andet sted i samme FAQ: i en organisation er roller

forholdsvis stabile, mens brugere og rettigheder er mange og skifter hurtigt, så når al adgang går gennem roller, bliver den enklere at administrere og at gennemgå.

Databaser implementerer ofte modellen direkte. PostgreSQL har siden version 8.1 kun haft én slags objekt til formålet, rollen, som kan fungere som bruger, som gruppe eller som begge dele. En rolle, der er tænkt som gruppe, har som regel ikke LOGIN, og medlemskab af den tildeles som enhver anden rettighed. To indstillinger for medlemskabet styrer, hvordan medlemmet bruger gruppens rettigheder: INHERIT giver medlemmet gruppens rettigheder automatisk, og SET lader medlemmet skifte over i rollen med SET ROLE, hvilket ligger tæt på RBAC-tanken om kun at aktivere en rolle i den session, der har brug for den.

```
-- rollen bærer rettighederne; ingen logger ind som den
CREATE ROLE reporting NOLOGIN;
GRANT USAGE ON SCHEMA sales TO reporting;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO reporting;

-- en person får rollen, aldrig tabellen
CREATE ROLE alex LOGIN;
GRANT reporting TO alex;

-- en magtfuld rolle, der skal slås til med vilje
CREATE ROLE schema_admin NOLOGIN;
GRANT schema_admin TO alex WITH INHERIT FALSE;
-- senere, i den ene session, der kræver det: SET ROLE schema_admin;
```

Brug begrebet i en beslutning

Brug »Hvad er RBAC?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Konti, adgangskoder og rettigheder sat op, så folk har præcis den adgang, jobbet kræver, og fratrådte mister den samme dag.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har skrevet 1.000 RBAC-regler til Linux-servere — 1,000 adgangsregler skrevet for at lukke et sikkerhedshul

Har skrevet 1.000 RBAC-regler til Linux-servere Kontakt os →

Hvorfor det betyder noget

Adgangskontrol skal besvare to spørgsmål, der trækker i hver sin retning: må denne anmodning gå igennem lige nu, og hvem kan hvad i hele systemet. De fleste ordninger klarer det første godt. RBAC er bygget til det andet, fordi svaret er en kort liste over roller og personerne i hver, som en revisor eller en leder

faktisk kan læse. Alternativet er attributbaseret adgangskontrol, hvor hver beslutning vurderer regler over egenskaber ved brugeren, ressourcen og konteksten, og NIST's dom over det er kort: det kan være let at sætte op, men komplekst at administrere.

Den praktiske beslutning er, hvor roller holder op med at slå til. Modellen fra 1992 giver adgang »kun hvis« rollekravene er opfyldt, en formulering valgt for at give plads til yderligere kontroller i selve adgangsøjeblikket. En lægerolle lader nogen læse journaler, men ikke enhver patients journal: forholdet mellem netop denne læge og netop denne patient er en kendsgerning om data, ikke en stillingsbetegnelse. Dér hører en anden mekanisme hjemme, for eksempel sikkerhed på rækkeniveau i databasen eller en attributregel i programmet, lagt oven på rollerne i stedet for at erstatte dem.

Vores anbefaling er at modellere adgang som roller først, opkaldt efter de job, folk faktisk har, med rettigheder knyttet udelukkende til roller og aldrig direkte til et login. Hold rollelisten så kort, at én person kan læse den hele. Tilføj regler på rækkeniveau eller attributregler kun dér, hvor en rolle beviseligt ikke kan trække grænsen, og lad databasen håndhæve alt, hvad den kan, for en kontrol i databasen gælder for alle klienter, også den kodesti, nogen glemte.

Sådan ser det ud i praksis

I en stilling som databaseingeniør fra 2022 til 2024 blev adgangen til hvert udviklermiljø, hver applikationsinstans, hver PostgreSQL- og MS SQL-database og hver Linux-server styret af 1.000 RBAC-regler. Modellen var den enkle del; mængden var udfordringen. Rettighederne blev delt op i genbrugelige kategorier som »ren læseadgang til produktionsdatabaser« og knyttet til roller som udviklere, applikationsinstanser og databaseadministratorer. Udviklere fik kun adgang til deres tildelte servere, og applikationsinstanser havde begrænsede rettigheder for at forhindre lateral bevægelse. Ansible anvendte reglerne ensartet på tværs af miljøer, og reglerne blev dokumenteret, så teams kunne gennemgå eller ændre dem. Det leverede resultat var det rollebaserede regelsæt og en gentagelig måde at anvende og gennemgå det på.

Roller arbejder sjældent alene. Da en kortapplikation flyttede fra MSSQL til PostgreSQL med PostGIS, blev rollebaseret adgangskontrol indført sammen med sikkerhed på rækkeniveau (row-level security) for at håndhæve

databeskyttelsespolitikker på tværs af flere brugergrupper. Den kombination er den sædvanlige arbejdsdeling: roller afgør, hvilke tabeller en gruppe må røre, og rækkepolitikker afgør, hvilke rækker i dem.

Den strengeste udgave fjerner tabellerne helt fra rækkevidde. I et funktionsbaseret datalag i PostgreSQL fra 2025 er hver handling, programmet udfører, en af 1.275 funktioner fordelt på 34 skemaer. Rollen, API'et logger ind som, mariner, har EXECUTE på programmets funktioner og USAGE på skemaerne og intet andet, hvilket giver omkring 4.000 eksplicitte tildelinger i stedet for én generel. Funktionerne kører som SECURITY DEFINER, ejet af en separat rolle uden login med fastlåst search_path, og superbrugerkontoen er forbeholdt migreringer og cron. Et kompromitteret API ville stadig være begrænset til det, funktionerne tillader. Prisen står der også: et nyt felt kræver en migrering og en ændring i en funktion.

Hvor det går galt

- **Rollerne formerer sig, til ingen kan overskue dem.** Hver undtagelse bliver en ny rolle, som »udvikler undtagen fakturering« eller »udvikler plus én rapport«, og snart er der flere roller end mennesker. Rolleeksplosion ophæver selve grunden til RBAC, nemlig en liste, der er kort nok til at blive gennemgået. Giv en person to eller tre roller i stedet for at oprette én pr. undtagelse, og slå roller sammen, når deres rettigheder er blevet identiske.
- **En rettighed havner direkte på et login.** Én hastesag, én GRANT direkte til en bruger, og den bruger har nu adgang, som en gennemgang af rollerne aldrig viser. Det er smuthullet, NIST beskriver i gruppesystemer, genopbygget i hånden. Tildel kun til roller, og lad gennemgangens SQL-forespørgsel lede efter rettigheder hos roller, der kan logge ind.
- **Standardindstillinger deler adgang ud, før nogen rolle gør det.** I PostgreSQL betyder navnet PUBLIC alle roller, og som standard får den EXECUTE på nye funktioner samt CONNECT og TEMPORARY på databaser. I databaser, der er opgraderet fra PostgreSQL 14 eller ældre, kan den også oprette objekter i skemaet public. Tilbagekald de standarder bevidst, helst i samme transaktion, som opretter objektet.
- **Roller vokser kun.** Folk skifter job og beholder den gamle rolle ved siden af den nye, og midlertidig adgang bliver permanent, fordi det ikke er nogens opgave at fjerne den. Hver tildeling skal have en begrundelse og en dato for næste gennemgang, og når nogen forlader et job, skal de roller, der fulgte med, forsvinde.

- **Nogle konti springer reglerne over.** Superbrugere og roller med BYPASSRLS ignorerer sikkerhed på rækkeniveau, og en tabels ejer gør det normalt også, medmindre tabellen er sat til FORCE ROW LEVEL SECURITY. Et program, der forbinder som ejeren eller som superbruger, gør hver rækkepolitik til pynt, så lad det forbinde som en rolle, der ikke ejer noget.

Læs mere

- NIST Role Based Access Control project — projektets arkiverede hjemmeside: modellens historie, standarden fra 2004 og revisionen fra 2012.
- NIST RBAC FAQ — roller over for grupper, RBAC0 til RBAC3, funktionsadskillelse og standardens fire komponenter.
- PostgreSQL: Role Membership — grupperoller, tildeling og tilbagekaldelse af medlemskab og valgmulighederne INHERIT og SET.
- PostgreSQL: Row Security Policies — standardafvisning, hvilke roller der går uden om rækkepolitikker, og FORCE ROW LEVEL SECURITY.
- PostgreSQL: Privileges — hvad PUBLIC får som standard, og hvordan det tilbagekaldes.
- SQL Server: Database-level roles — faste og brugerdefinerede roller i en SQL Server-database.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er RBAC?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/rbac/