Engineer

RBAC

tillatelser gitt til en rolle, ikke til en person

Hva det er

RBAC avgjør hva noen får gjøre ut fra rollene de har, og ikke ut fra hvem de er. Bokstavene står for role-based access control, rollebasert tilgangskontroll, og NIST, der modellen ble formalisert, kaller det også «role based security». En rolle er en navngitt samling tillatelser, f.eks. «regnskapsmedarbeider» eller «skrivebeskyttet i produksjon». Brukere tildeles roller, og hver tillatelse en bruker utøver, kommer gjennom én av dem.

David Ferraiolo og Rick Kuhn ved NIST foreslo den generelle modellen i 1992, som et alternativ til datidens obligatoriske og skjønnsbaserte tilgangskontroll. I 1996 delte Sandhu, Coyne, Feinstein og Youman den opp i lag: RBAC0 med brukere, roller og tillatelser, RBAC1 som legger til rollehierarkier, RBAC2 som legger til begrensninger, og RBAC3 med begge. De to linjene ble slått sammen og ble standarden ANSI/INCITS 359-2004, revidert i 2012. Referansemodellen i standarden har fire komponenter. Kjerne-RBAC (Core RBAC), den eneste hvert system må ha, holder tildelingene av brukere til roller og av tillatelser til roller, pluss økten, der en bruker aktiverer et utvalg av rollene den er tildelt. Hierarkisk RBAC lar en overordnet rolle arve tillatelsene til en underordnet. Statisk ansvarsdeling (separation of duty) begrenser hvilke roller én bruker kan tildeles samtidig, og dynamisk ansvarsdeling begrenser hvilke roller som kan være aktive samtidig i én økt, så noen kan ha både rollen som oppretter en betaling og rollen som godkjenner den, uten å bruke begge i samme økt.

Forskjellen fra en vanlig gruppe er mindre enn den ser ut. FAQ-en til NIST sier det slik: en gruppe er som regel en samling brukere, og de fleste gruppemekanismer lar også en tillatelse knyttes rett til en bruker, noe FAQ-en kaller et smutthull (loophole) som gjør forholdet mellom brukere og tillatelser vanskelig å styre. RBAC tetter det ved å knytte tillatelser bare til roller. Grunnen står i en annen

linje i den samme FAQ-en: roller i en organisasjon er relativt stabile, mens brukere og tillatelser er mange og endrer seg raskt, så når all tilgang går gjennom roller, blir den enklere å forvalte og å gå gjennom.

Databaser implementerer ofte modellen direkte. PostgreSQL har siden versjon 8.1 hatt én type entitet, rollen, som kan opptre som en bruker, en gruppe eller begge deler. En rolle som er ment som gruppe, har som regel ikke LOGIN, og medlemskap i den tildeles som et hvilket som helst annet privilegium. To innstillinger for medlemskapet styrer hvordan medlemmet bruker gruppens privilegier: INHERIT gir medlemmet privilegiene til gruppen automatisk, og SET lar medlemmet bytte til rollen med SET ROLE, noe som ligger nær RBAC-tanken om å aktivere en rolle bare i økten som trenger den.

```
-- 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;
```

Bruk begrepet i en beslutning

Bruk «Hva er RBAC?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Kontoer, passord og tilganger ordnet slik at folk har nøyaktig den tilgangen jobben krever, og de som slutter, mister den samme dag.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Skrev 1 000 RBAC-regler for å sikre Linux-servere — 1,000 tilgangsregler skrevet for å tette et sikkerhetshull

Skrev 1 000 RBAC-regler for å sikre Linux-servere Ta kontakt →

Hvorfor det er viktig

Tilgangskontroll må svare på to spørsmål som drar i hver sin retning: er denne forespørselen tillatt akkurat nå, og hvem kan gjøre hva i hele systemet. De fleste ordninger svarer godt på det første. RBAC er bygd for det andre, fordi svaret er en kort liste over roller og personene i hver av dem, som en revisor eller en leder

faktisk kan lese. Alternativet er attributtbasert tilgangskontroll (ABAC), der hver beslutning evaluerer regler over attributter ved brukeren, ressursen og konteksten, og NIST oppsummerer det med at det kan være enkelt å sette opp, men komplisert å forvalte.

Den praktiske beslutningen er hvor roller slutter å strekke til. Modellen fra 1992 gir tilgang «bare hvis» (only if) kravene til rollen er oppfylt, en formulering som ble valgt for å gi rom for flere sjekker i det øyeblikket tilgangen skjer. En legerolle lar noen lese pasientjournaler, men ikke hver pasients journal: forholdet mellom denne legen og den pasienten er et faktum om dataene, ikke en stillingstittel. Det er der en annen mekanisme hører hjemme, som sikkerhet på radnivå i databasen eller en attributtregel i applikasjonen, lagt oppå rollene i stedet for å erstatte dem.

Vår anbefaling er å modellere tilgang som roller først, oppkalt etter jobber folk faktisk gjør, med tillatelser knyttet bare til roller og aldri direkte til en innlogging. Hold rollelisten kort nok til at én person kan lese hele. Legg til regler på radnivå eller attributtregler bare der en rolle påviselig ikke kan trekke grensen, og la databasen håndheve det den kan, fordi en sjekk databasen gjør, gjelder for hver klient, også kodeveien noen glemte.

Slik ser det ut i praksis

I en jobb med databaseutvikling fra 2022 til 2024 ble tilgangen til hvert utviklermiljø, hver applikasjonsinstans, hver database i PostgreSQL og MS SQL og hver Linux-server styrt av 1 000 RBAC-regler. Modellen var den enkle delen; volumet var utfordringen. Tillatelsene ble delt inn i gjenbrukbare kategorier som «skrivebeskyttet tilgang til produksjonsdatabaser» og knyttet til roller som utviklere, applikasjonsinstanser og databaseadministratorer. Utviklere kunne bare nå serverne som var tildelt dem, og applikasjonsinstanser hadde begrensede tillatelser for å hindre lateral bevegelse. Ansible anvendte reglene konsekvent på tvers av miljøer, og reglene ble dokumentert slik at teamene kunne revidere eller endre dem. Leveransen var det rollebaserte regelsettet og en gjentakbar metode for å anvende og gjennomgå det.

Roller jobber sjelden alene. Da en kartapplikasjon flyttet fra MSSQL til PostgreSQL med PostGIS, kom rollebasert tilgangskontroll inn sammen med sikkerhet på radnivå for å håndheve retningslinjer for databeskyttelse på tvers av

flere brukergrupper. Den kombinasjonen er den vanlige arbeidsdelingen: rollene avgjør hvilke tabeller en gruppe kan røre, og radreglene avgjør hvilke rader inne i dem.

Den strengeste varianten tar tabellene helt ut av rekkevidde. I et datalag i PostgreSQL bygd med funksjonene først fra 2025 er hver operasjon applikasjonen utfører, én av 1 275 funksjoner fordelt på 34 skjemaer. Rollen API-et logger inn som, mariner, har EXECUTE på applikasjonsfunksjonene og USAGE på skjemaene og ingenting annet, noe som blir omtrent 4 000 eksplisitte tildelinger i stedet for én generell. Funksjonene kjører som SECURITY DEFINER, eid av en egen rolle uten innlogging og med fast search_path, og superbrukerkontoen er forbeholdt migreringer og cron. Et kompromittert API ville fortsatt være begrenset til det funksjonene tillater. Prisen er også skrevet ned: et nytt felt betyr en databasemigrering og en funksjonsendring.

Hvor det går galt

- **Rollene formerer seg til ingen kan lese dem.** Hvert unntak blir en ny rolle, som «utvikler unntatt fakturering» eller «utvikler pluss én rapport», og snart finnes det flere roller enn personer. En slik rolleeksplosjon (role explosion) undergraver selve grunnen til RBAC, som var en liste kort nok til å gå gjennom. Gi en person to eller tre roller i stedet for å lage én per unntak, og slå sammen roller som har endt opp med identiske sett tillatelser.
- **En tillatelse havner rett på en innlogging.** Én hastesak, én GRANT rett til en bruker, og den brukeren har nå tilgang som ingen gjennomgang av roller vil vise. Det er smutthullet NIST beskriver i gruppesystemer, bygd opp igjen for hånd. Tildel bare til roller, og la spørringen i gjennomgangen lete etter privilegier som holdes av roller som kan logge inn.
- **Standardverdier deler ut tilgang før noen rolle gjør det.** I PostgreSQL betyr navnet PUBLIC hver rolle, og som standard får den EXECUTE på nye funksjoner og CONNECT og TEMPORARY på databaser. I databaser som er oppgradert fra PostgreSQL 14 eller eldre, kan den også opprette objekter i skjemaet som heter public. Trekk tilbake disse standardene med vilje, helst i den samme transaksjonen som oppretter objektet.
- **Rollene bare vokser.** Folk bytter jobb og beholder den gamle rollen ved siden av den nye, og midlertidig tilgang blir permanent fordi det ikke er noens oppgave å fjerne den. Hver tildeling trenger en begrunnelse og en dato for gjennomgang, og den som slutter i en jobb, bør miste rollene som fulgte med den.

- **Noen kontoer hopper over reglene.** Superbrukere og roller med BYPASSRLS ser bort fra sikkerhet på radnivå, og det gjør som regel også eieren av en tabell, med mindre tabellen er satt til FORCE ROW LEVEL SECURITY. En applikasjon som kobler til som eieren eller som superbruker, gjør hver radregel til pynt, så la den koble til som en rolle som ikke eier noe.

Les mer

- NIST Role Based Access Control project — det arkiverte prosjektet: historien til modellen, standarden fra 2004 og revisjonen fra 2012.
- NIST RBAC FAQ — roller mot grupper, RBAC0 til RBAC3, ansvarsdeling og de fire komponentene i standarden.
- PostgreSQL: Role Membership — grupperoller, tildeling og tilbaketrekking av medlemskap, og valgene INHERIT og SET.
- PostgreSQL: Row Security Policies — avvisning som standard, hvilke roller som går forbi radreglene, og FORCE ROW LEVEL SECURITY.
- PostgreSQL: Privileges — hva PUBLIC får som standard, og hvordan du trekker det tilbake.
- SQL Server: Database-level roles — faste og brukerdefinerte roller inne i en database i SQL Server.

Ordliste

Velg ut fra deres faktiske rammer

Ta med utrullingsløpet, driftsrisikoene og dem som har vakt. Vi kan avgrense en endring med kontroller, en vei tilbake og tydelig ansvar.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er RBAC?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/rbac/