Engineer

RBAC

behörigheter som ges till en roll, inte till en person

Vad det är

RBAC avgör vad någon får göra genom att titta på de roller personen har i stället för på vem personen är. Bokstäverna står för role-based access control, rollbaserad åtkomstkontroll, och NIST, där modellen formaliserades, kallar den också ”role based security”. En roll är en namngiven samling behörigheter, t.ex. ”reskontraansvarig” eller ”läsbehörighet i produktion”. Användare tilldelas roller, och varje behörighet som en användare utövar kommer via någon av dem.

David Ferraiolo och Rick Kuhn vid NIST föreslog den allmänna modellen 1992, som ett alternativ till den obligatoriska och den skönsmässiga åtkomstkontroll som var vanlig då. År 1996 delade Sandhu, Coyne, Feinstein och Youman upp den i lager: RBAC0 med användare, roller och behörigheter, RBAC1 som lägger till rollhierarkier, RBAC2 som lägger till begränsningar och RBAC3 med båda. De två linjerna förenades och blev standarden ANSI/INCITS 359-2004, reviderad 2012. Dess referensmodell har fyra komponenter. Kärn-RBAC, den enda som varje system måste ha, rymmer kopplingarna mellan användare och roller och mellan behörigheter och roller, plus sessionen, där en användare aktiverar en delmängd av de roller som tilldelats. Hierarkisk RBAC låter en överordnad roll ärva en underordnad. Statisk åtskillnad av arbetsuppgifter begränsar vilka roller en användare får tilldelas samtidigt, och dynamisk åtskillnad begränsar vilka roller som får vara aktiva på en gång i en session. Någon kan alltså ha både rollen som registrerar en betalning och rollen som godkänner den, utan att använda båda vid samma tillfälle.

Skillnaden mot en vanlig grupp är mindre än den ser ut. NIST förklarar det så i sina vanliga frågor och svar: en grupp är oftast en samling användare, och de flesta gruppmekanismer låter också en behörighet knytas direkt till en användare, något som NIST kallar ett ”loophole”, ett kryphål som gör relationen mellan användare och behörigheter svår att styra. RBAC stänger det genom att bara knyta behörigheter till roller. Skälet står på en annan rad i samma text: roller

i en organisation är relativt stabila, medan användare och behörigheter är många och ändras snabbt, så att leda all åtkomst genom roller gör den enklare att hantera och att granska.

Databaser implementerar ofta modellen direkt. PostgreSQL har sedan version 8.1 haft en enda sorts entitet, rollen, som kan fungera som användare, som grupp eller som båda. En roll som är tänkt som grupp har oftast ingen LOGIN, och medlemskap i den beviljas som vilket annat privilegium som helst. Två medlemskapsinställningar styr hur en medlem använder gruppens privilegier: INHERIT ger medlemmen gruppens privilegier automatiskt, och SET låter medlemmen byta till rollen med SET ROLE, vilket ligger nära RBAC-idén om att bara aktivera en roll i den session som behöver den.

```
-- rollen bär behörigheterna; ingen loggar in 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 kraftfull roll som måste slås på med avsikt
CREATE ROLE schema_admin NOLOGIN;
GRANT schema_admin TO alex WITH INHERIT FALSE;
-- senare, i den enda session som behöver den: SET ROLE schema_admin;
```

Använd begreppet i ett beslut

Använd ”Vad är RBAC?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Konton, lösenord och behörigheter ordnade så att alla har exakt den åtkomst deras arbete kräver, och de som slutar förlorar den samma dag.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Skrev 1 000 RBAC-regler som säkrar Linux-servrar — 1,000 åtkomstregler, skrivna för att täppa till en säkerhetslucka

Skrev 1 000 RBAC-regler som säkrar Linux-servrar Hör av er →

Varför det spelar roll

Åtkomstkontroll måste besvara två frågeställningar som drar åt olika håll: är den här begäran tillåten just nu, och vem kan göra vad i hela systemet. De flesta upplägg besvarar den första väl. RBAC är byggt för den andra, eftersom svaret är en kort lista med roller och personerna i varje, som en revisor eller en chef

faktiskt kan läsa. NIST sammanfattar alternativet, attributbaserad åtkomstkontroll, där varje beslut prövar regler mot attribut hos användaren, resursen och sammanhanget, med orden att det ”can be easy to set up but complex to manage”: lätt att sätta upp men komplext att förvalta.

Det praktiska beslutet gäller var rollerna slutar räcka. Modellen från 1992 ger åtkomst ”only if”, endast om, rollens krav är uppfyllda, en formulering som valdes för att lämna utrymme för fler kontroller vid själva åtkomsten. En läkarroll låter någon läsa journaler, men inte varje patients journal: relationen mellan just den här läkaren och den patienten är ett faktum om datan, inte en yrkestitel. Där hör en andra mekanism hemma, som säkerhet på radnivå i databasen eller en attributregel i applikationen, lagd ovanpå rollerna i stället för att ersätta dem.

Vår rekommendation är att först modellera åtkomst som roller, uppkallade efter jobb som människor faktiskt utför, med behörigheter knutna enbart till roller och aldrig direkt till en inloggning. Håll rollistan så kort att en person kan läsa hela. Lägg bara till regler på radnivå eller attributregler där en roll bevisligen inte kan dra gränsen, och låt databasen upprätthålla allt den kan, eftersom en kontroll som databasen gör gäller för varje klient, även den kodväg som någon glömde.

Så ser det ut i praktiken

I ett jobb med databasteknik mellan 2022 och 2024 styrdes åtkomsten till varje utvecklarmiljö, applikationsinstans, PostgreSQL- och MS SQL-databas och Linux-server av 1 000 RBAC-regler. Modellen var den enkla delen; volymen var utmaningen. Behörigheterna delades upp i återanvändbara kategorier som ”skrivskyddad åtkomst till produktionsdatabaser” och knöts till roller som utvecklare, applikationsinstanser och databasadministratörer. Utvecklare nådde bara sina tilldelade servrar, och applikationsinstanserna hade begränsade behörigheter för att förhindra lateral förflyttning. Ansible tillämpade reglerna enhetligt i miljöerna, och reglerna dokumenterades så att teamen kunde granska eller ändra dem. Det levererade resultatet var den rollbaserade regeluppsättningen och ett repeterbart sätt att tillämpa och granska den.

Roller arbetar sällan ensamma. När en kartapplikation flyttades från MSSQL till PostgreSQL med PostGIS infördes rollbaserad åtkomstkontroll tillsammans med säkerhet på radnivå för att upprätthålla dataskyddspolicyer för flera användargrupper. Det paret är den vanliga arbetsfördelningen: roller avgör vilka tabeller en grupp får röra, och radpolicyer avgör vilka rader inuti dem.

Den striktaste varianten tar tabellerna helt ur räckhåll. I ett funktionsbaserat datalager i PostgreSQL från 2025 är varje operation som applikationen utför en av 1 275 funktioner fördelade på 34 scheman. Rollen som API:et loggar in som, mariner, har EXECUTE på applikationens funktioner och USAGE på schemana och inget annat, vilket blir ungefär 4 000 explicita behörigheter i stället för en generell. Funktionerna körs som SECURITY DEFINER och ägs av en separat roll utan inloggning med en fast search_path, och superanvändarkontot är reserverat för migreringar och cron. Ett komprometterat API skulle fortfarande vara begränsat till det som funktionerna tillåter. Priset står också nedskrivet: ett nytt fält innebär en databasmigrering och en ändrad funktion.

Var det går fel

- **Rollerna förökar sig tills ingen kan läsa dem.** Varje undantag blir en ny roll, som ”utvecklare utom fakturering” eller ”utvecklare plus en rapport”, och snart finns det fler roller än människor. Rollexplosionen motverkar själva skälet till RBAC, som var en lista kort nog att granska. Ge en person två eller tre roller i stället för att skapa en ny per undantag, och slå ihop roller vars uppsättningar av behörigheter har blivit identiska.
- **En behörighet hamnar direkt på en inloggning.** Ett brådskande ärende, en GRANT direkt till en användare, och den användaren har nu åtkomst som ingen granskning av roller visar. Det är kryphålet som NIST beskriver i gruppsystem, återuppbyggt för hand. Bevilja bara till roller, och låt den databasfråga som granskar leta efter privilegier hos roller som kan logga in.
- **Standardinställningar delar ut åtkomst innan någon roll gör det.** I PostgreSQL betyder namnet PUBLIC varje roll, och som standard får den EXECUTE på nya funktioner och CONNECT och TEMPORARY på databaser. I databaser som har uppgraderats från PostgreSQL 14 eller tidigare kan den också skapa objekt i schemat som heter public. Återkalla de standardinställningarna med avsikt, helst i samma transaktion som skapar objektet.
- **Roller växer bara.** Människor byter jobb och behåller den gamla rollen bredvid den nya, och tillfällig åtkomst blir permanent eftersom det inte är någons uppgift att ta bort den. Varje tilldelning behöver ett skäl och ett datum för granskning, och den som lämnar ett jobb ska förlora de roller som hörde till det.
- **Vissa konton hoppar över reglerna.** Superanvändare och roller med BYPASSRLS ignorerar säkerhet på radnivå, och det gör normalt även tabellens ägare, om inte tabellen är inställd på FORCE ROW LEVEL SECURITY. En applikation som ansluter som ägaren eller som superanvändare gör varje radpolicy till dekoration, så låt den ansluta som en roll som inte äger något.

Läs mer

- NIST Role Based Access Control project — projektets arkiverade startsida: modellens historia, standarden från 2004 och revisionen från 2012.
- NIST RBAC FAQ — roller jämfört med grupper, RBAC0 till RBAC3, åtskillnad av arbetsuppgifter och standardens fyra komponenter.
- PostgreSQL: Role Membership — grupproller, att bevilja och återkalla medlemskap och alternativen INHERIT och SET.
- PostgreSQL: Row Security Policies — standardläget att neka, vilka roller som går förbi radpolicyer och FORCE ROW LEVEL SECURITY.
- PostgreSQL: Privileges — vad PUBLIC får som standard och hur man återkallar det.
- SQL Server: Database-level roles — fasta och användardefinierade roller inne i en databas i SQL Server.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är RBAC?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/rbac/