IN2090 – Databaser og datamodellering

12 – Sikkerhet i databaser

Leif Harald Karlsen
leifhka@ifi.uio.no

Universitetet i Oslo

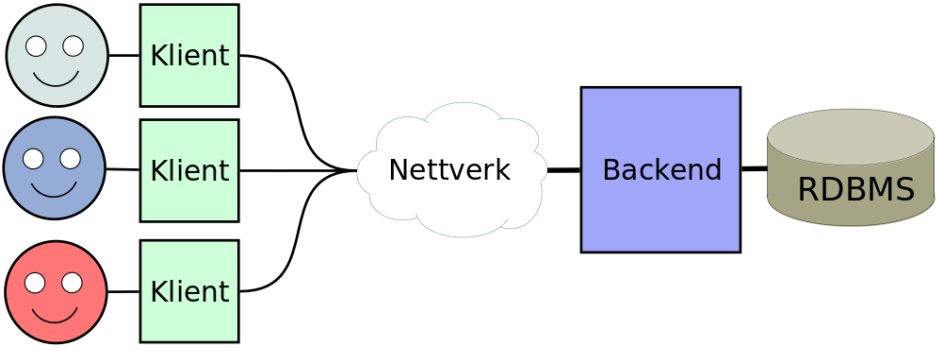
Hovedmål med databasesikkerhet

- Konfidensialitet
  - Uvedkommende må ikke kunne se data de ikke skal ha tilgang til
- Intergritet
  - Data må være korrekte og pålitelige. Derfor må data beskyttes mot endringer fra uautoriserte brukere
- Tilgjengelighet
  - Brukere må kunne se eller modifiesere data de har fått tilgang til

Sikkerhet: Ikke bare i databasesystemet

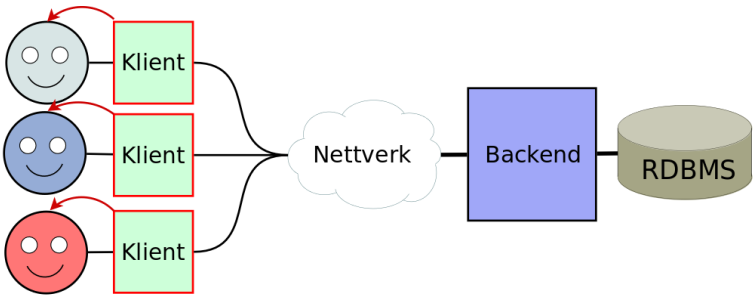
- Programmer inneholder ofte mye mer enn bare databasen
- Databasesikkerhet kan derfor ikke kun fokusere på databasen
- Sikkerhetshull kan forekomme i alle ledd (frontend, backend, nettverket, osv.)
- Sikkerhet er derfor alltid en *helhetlig* oppgave
- Må derfor sikre hver enkelt komponent og interaksjonen mellom dem
- Må være tydelige på hvilke antagelser om andre komponenter hver del av systemet gjør

Oversikt over systemer med databaser



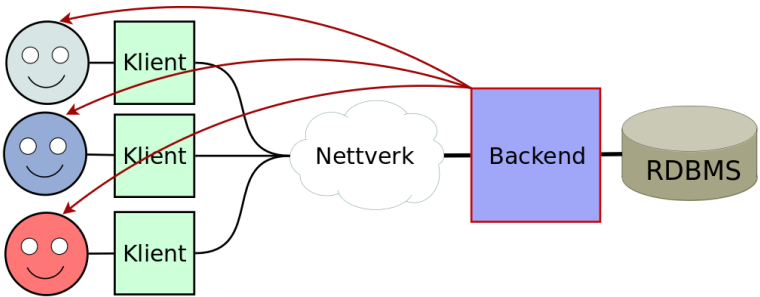
Tilgangskontroll: Klient/Frontend

- Klienten autentiserer brukerne
- Klienten sjekker hva brukeren har lov til
- Backend og RDBMS må stole på at klienten gjør dette riktig
- Trenger sikring slik at bare klienten kan få tilgang til backend/RDBMS



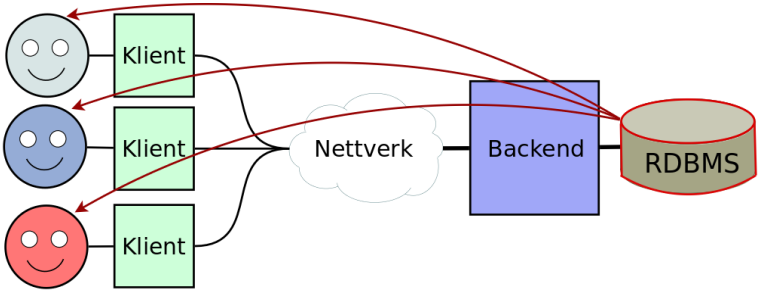
Tilgangskontroll: Backend

- Backend autentiserer brukerne
- Backend sjekker hva brukeren har lov til
- Backend har ofte én bruker til databasen
- RDBMS må stole på at backend gjør dette riktig
- Backend og RDBMS ofte bak samme brannmur



Tilgangskontroll: Database

- Databasesystemet autentiserer brukerne direkte
- Hver bruker av programmet får da hver sin databasebruker
- Klienten kan forhåndssjekke (f.eks. for å tilpasse brukergrensesnittet)



Tilgangskontroll i databaser

- Tilgang til databasen kontrolleres gjennom tre ting:
  - brukere
  - roller
  - rettigheter
- For eksempel:
  - Bruker `leifhka` har rollen `kundeadmin`
  - Bruker `leifhka` har rollen `produktansvarlig`
  - Bruker `klient` har rollen `kunde`
  - Brukere med rollen `kundeadmin` kan opprette og oppdatere kunder (rader i `customer`-tabellen)
  - Brukere med rollen `produktansvarlig` kan opprette, oppdatere og slette produkter (rader i `products`-tabellen)
  - Brukere med rollen `kunde` kan se på produkter (rader i `products`-tabellen) samt legge inn ordre (rader i `orders`-tabellen)

Brukere vs. roller

- Mulig å gi hver bruker de rettighetene de skal ha
- Men vanskelig å holde rede på at hver bruker har de riktige rettighetene
- Spesielt om det er mange brukere og mange rettigheter
- Typisk vil mange brukere trenge samme rettigheter: Vanskelig å vedlikeholde
- Vi lager derfor roller som fanger en mengde med rettigheter som hører sammen
- Og gir deretter brukere de passende rollene
- Dette heter *Role-based Access Control*

Databasebrukere

- F.eks. når dere logger dere inn i databasen med:

  ```
  $ psql -h dbpg-ifi-kurs01 -U leifhka -d fdb
  ```

  er `leifhka` brukeren
- Autentisering skjer typisk via passord, SSH public keys, el.
- Gydige brukernavn og (krypterte) passord lagres av RDBMS
- Autentisering kan delegeres til andre systemer
- Alle databaser, skjema, tabeller, views, osv. eies av en bruker

Lage brukere og roller med SQL

- For å lage en ny bruker `leifhka` med passord `hemmelig` og rollene `kundeadmin` og `produktansvarlig` kan man kjøre følgende SQL-kommando[1]

```
CREATE USER leifhka WITH PASSWORD 'hemmelig'
    ROLE kundeadmin, produktansvarlig;
```

- Roller lages nesten helt likt[2]:

```
CREATE ROLE produktansvarlig;
```

- I PostgreSQL er `CREATE USER` bare et alias for `CREATE ROLE` med `LOGIN`-adgang (mao. brukere er bare en spesiell type rolle)
- Roller og brukre slettes med `DROP`
- Merk: Som oftest bare superbrukere som kan lage brukere/roller

[1] se `https://www.postgresql.org/docs/12/sql-createuser.html`
[2] se `https://www.postgresql.org/docs/12/sql-createrole.html`

Begrense bruk

- Kan bergrense hvor lenge en bruker eller rolle skal være gyldig ved å sette `VALID UNTIL '2021-01-01'` i kommandoene over
- Kan begrense antall tilkoblinger en bruker/rolle kan ha ved å sette `CONNECTION LIMIT 5` i kommandoene over
- Dette er det som gjør at noen av dere har fått feilmeldingen:

```
psql: FATAL:  too many connections for role "user_name"
```

- For å gi en bruker/rolle (generelle) rettigheter til å lage databaser, roller, osv. kan man legge til `CREATEDB`, `CREATEROLE`, osv.

Gi og fjerne rettigheter

- Man kan gi roller/brukere mer detaljerte rettigheter via `GRANT`-kommandoen[3]
- `GRANT`-kommandoen har følgende form:

```
GRANT <privileges> ON <object> TO <role>;
```

- hvor `<privileges>` f.eks.:

```
SELECT, UPDATE, INSERT, DELETE, CREATE, CONNECT, USAGE, ALL
```

- og `<object>` er f.eks. en database, en tabell, et skjema, el.
- Gir man rettigheter til en rolle, vil alle dens medlemmer også få disse
- Fjerning av rettigheter kan gjøres tilsvarende med `REVOKE`

[3] `https://www.postgresql.org/docs/12/sql-grant.html`

GRANT-eksempler

- For å gi rollen `kundeadmin` rettighetene til å oprette og oppdatere `ws.users`-tabellen kan vi kjøre følgende kommando:

```
GRANT INSERT, UPDATE ON TABLE ws.users TO kundeadmin;
```

- For å gi rollen `webshopadmin` alle rettigheter innenfor skjemaet `ws`:

```
GRANT ALL ON SCHEMA ws TO webshopadmin;
```

- Kan også gi en bruker en ny rolle med `GRANT`:

```
GRANT kundeadmin TO leifhka;
```

- Kan til og med gi tillatelser på kolonnenivå:

```
GRANT UPDATE (price) ON ws.products TO prisansvarlig;
```

- For å fjerne `kunde`-rollens tilgang til `categories` kan vi kjøre

```
REVOKE USAGE ON ws.categories FROM kunde;
```

Tilgang og views

- I enkelte tilfeller ønsker vi ikke gi tilgang til tabellene direkte
- Men f.eks. kun aggregerte eller utvalgte verdier
- F.eks. vil ikke gi tilgang til pasientjournalen til hver enkelt person, men heller antall med ulike sykdommer per kommune
- Kan da lage views, og så gi tilgang til disse

Takk for nå!

Neste video handler om sikkerhet i programmer som bruker databaser.