Beveiliging

Wim.bertels@ucll.be

Security

- Hardware
- Het platform waar het op draait
- De databank software
- Binnen sql zelf (grant, revoke, view,..)
- Sql als “vertaler”
- Applicaties
- Gebruikers
- Algemeen terugkerende security problematiek (bv CIA, AAA, maar ook reserve kopies, ..)
- ..

Ondersteunend platform

- Bijdetijds (Up to date) houden
- Beveiligen (infrastructuur vakken)
- (D)DoS-attack's
- ..

Databank software

- Up to date houden
- Configuratie
  - Local
  - Remote
    - Eg postgresql: pg_hba.conf, postmaster.conf en postgresql.conf
- Known exploits -> Wat doe je?

Binnen SQL zelf

- User management (roles)
- Privileges (grant / revoke)
- Views
- Stored procedures
- Row Security Policies

SQL

- SQL wordt niet gecompileerd
- SQL wordt “vertaald” in de onderliggende/omsluitende laag
- Dit kan gebruikt worden om sql ongewenste instructies te laten uitvoeren
- Het is probleem dat bij elke taal die “vertaald”, terugkomt (eg php)

SQL Injection

- Indien we de onderliggende sql code van bv een formulier niet kennen, dan gaan we proberen te raden wat de sql code is die erachter zit.
- Ipv gewone waarden gaan sql code meegeven met het formulier.
- Pas op serial, identity, autoincrement..?

Incorrectly Filtered Escape Characters

- Fout: input is niet gefilterd op escape characters
- Bv
  - statement := "SELECT * FROM users WHERE naam = '" + userNaam + "';"
  - UserNaam:= a' or 't'='t
  - Gevolg: SELECT * FROM users WHERE naam = 'a' or 't'='t';
  - Dus.. , andere voorbeelden?

Incorrect Type Handling

- Fout: types van de input worden niet gecheckt
- Bv
  - statement := "SELECT * FROM data WHERE id = " + a + ";" (a wordt enkel verwacht als int)
  - Voor a := 1;DROP TABLE users;
  - Gevolg: SELECT * FROM data WHERE id = 1;DROP TABLE users;
  - Dus..

Oplossingen

- Escape character verwijderen uit de invoervelden
- Invoer validatie, controleren of het invoerveld bv wel het juiste type heeft
- Prepared statements
- Stored procedures
  - Type
  - ; en escaping
  - Grants..
  - Dicht bij de bron
  - ..

Opgepast

- Enkel Escaping is niet voldoende:
- Bv
  - SELECT * from items where userid=$userid;
  - $userid := "33 or userid is not null or userid=44";
  - SELECT * from items where userid=33 or userid is not null or userid=44;
- Dus zeg eerder wat wel mag zijn als invoer,ipv wat niet mag; het eerste is eenvoudiger, er zijn (meestal) maar een eindig aantal mogelijkheden.

Handige Functies

- quote_ident (text) → text
- quote_literal (anyelement) → text
  - quote_nullable (anyelement) → text

Dieper

- https://www.postgresql.org/docs/current/ddl-rowsecurity.html

```
CREATE TABLE users (
        manager text,
        company text );
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
CREATE POLICY users_manager
        ON users TO managers  -- groep rol managers
        USING (manager = current_user);
```

- https://www.postgresql.org/docs/current/sepgsql.html
- https://www.postgresql.org/docs/current/sql-security-label.html

Links

- http://www.w3schools.com/
- http://www.postgresql.org/
- http://en.wikipedia.org/