Input Validation 2

Zoals altijd kunnen de oefeningen binnengehaald worden door een git pull te doen.

```
git pull
```

of de git repository te clonen als je deze nog niet hebt.

```
git clone https://github.com/similonap/software_security_2021.git
```

Wat ga je leren in dit labo?

- Uitvoeren van SQL Injection aanvallen
- Oplossen van SQL Injection risico's

Database design

De applicatie bestaat uit twee tabellen:

```
CREATE TABLE Ticket (name STRING, country STRING, tickets INTEGER)
CREATE TABLE User (username STRING, password STRING, admin BOOLEAN, fullname STRING)
```

Stappenplan

1. Ga naar de `labo_validation` directory en doe vervolgens

   ```
   npm install
   ```

2. Hier vinden we weer onze onveilige web applicatie! Je kan deze opstarten net als vorig labo met

   ```
   node index.js
   ```

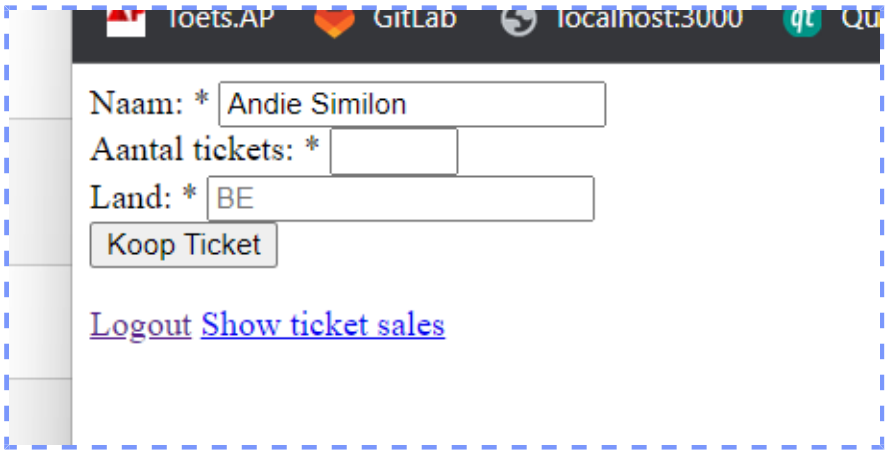
3. Probeer een SQL injection aanval op het login scherm. Je kan het Username veld misbruiken om jezelf in te loggen als admin.

   Tip: De SQL Query wordt afgeprint in je console venster. Zo kan je gemakkelijker zien welke query er uitgevoerd wordt bij het inloggen.

Welke string heb je moeten ingeven om de SQL aanval uit te voeren:

```
admin'--
```

Neem een screenshot van de admin pagina en sleep deze hieronder in



4. Bij show ticket sales kun je een overzicht krijgen van alle gekochte tickets. Je kan daar ook zoeken op alle tickets. De query die hier gebruikt wordt is ook gevoelig aan een SQL injection aanval.

5. Het is mogelijk om alle tabellen van de database uit te printen aan de hand van een UNION gebaseerde aanval.

   In SQLite kan je alle tabellen opvragen via

   ```
   SELECT name FROM sqlite_master WHERE type ='table';
   ```

   Je kan in het search input veld via de ticket sales pagina een SQL injection aanval doen door:

   ```
   %' UNION SELECT name FROM sqlite_master WHERE type ='table'; --
   ```

   in te geven in het invoerveld. Dit werkt jammer genoeg niet, waarom niet?

   ```
   het aantal kolommen komt niet overeen
   ```

   Tip: vergelijk het aantal kolommen van de query hier boven en de query die gedaan wordt om de Tickets op te vragen.

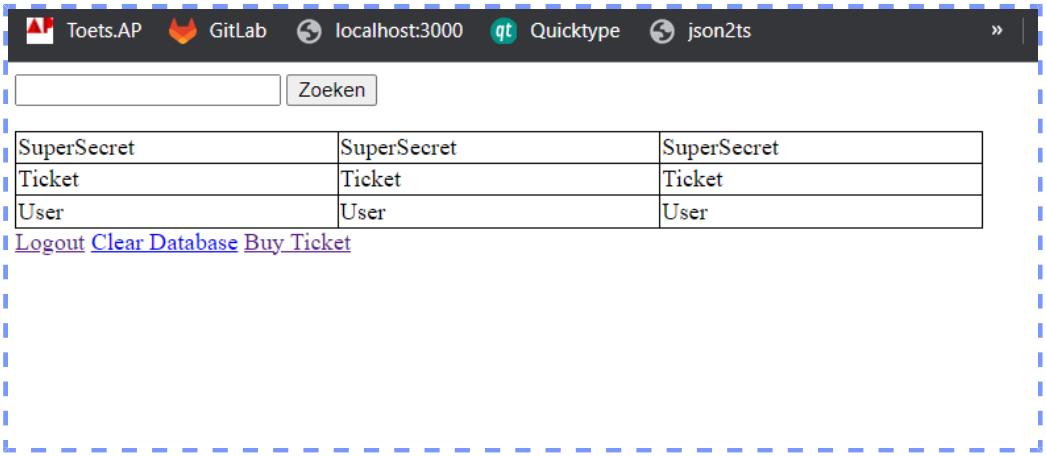
6. Pas nu de query string van hierboven aan zodat er evenveel kolommen geselecteerd worden als de query die wordt gedaan voor show-tickets.

Tip: Je kan lege kolommen toevoegen a.d.h.v '' of gewoon de naam meerdere keren selecteren.

Welke string heb je moeten ingeven om de SQL aanval uit te voeren:

```
ik ben blij dat ik de tips nooit per ongeluk lees.
%' UNION SELECT name, name, name FROM sqlite_master WHERE type ='table'; --
```

Neem een screenshot van de uitgeprinte tabellen en sleep deze hieronder in:

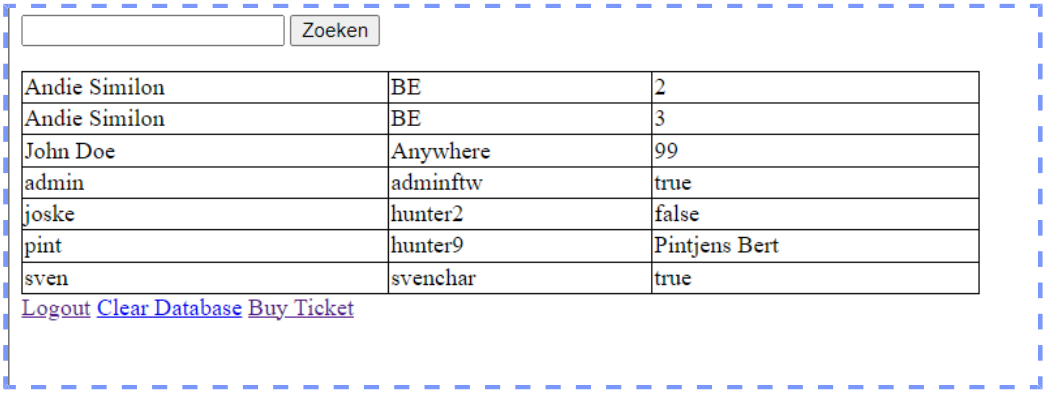


7. Je kan aan de hand van een UNION sql injection aanval ook de User tabel uitlezen.

   Welke string heb je moeten ingeven om de SQL aanval uit te voeren:

   ```
   %' UNION SELECT username, password, admin FROM user; --
   ```

   Neem een screenshot van de uitgeprinte User tabel en sleep deze hieronder in:



8. **Extra:** Ook bij het kopen van tickets kan je een SQL Injection aanval doen. Probeer een User toe te voegen door het `Naam` veld te misbruiken.

   Zorg er eerst voor dat je de wijzigingen voor server side validation van vorige les in commentaar zet!

   Wat moest je in het Naam veld ingeven om een User toe te voegen.

```
John Doe', 'Anywhere', 99) ; INSERT INTO User VALUES ("pint", "hunter9", 'Pintjens Bert',
'true')--
```

9. Gebruik prepared statements om de applicatie veilig te maken voor SQL injection attacks.

   Opgepast: Zorg dat alles blijft werken. Bij de LIKE query zal je '%' + name + '%' moeten meegeven als parameter voor het vraagteken.

10. Print deze pagina af als PDF en slaag deze op als `naam_voornaam_labo_sql_injection.pdf` en stuur de volgende files op via digitap:

    - naam_voornaam_labo_sql_injection.pdf
    - tickets.db
    - index.js