Week-11 A1 SQL Injection

- **Give an example of a SQL inject which will give all users in a user table**
  Brug Juice Shop igen.

De har et rest endpoint på `/rest/products/search?q=`
Indsæt nedenstående ind efter q=

```
qwert')) UNION SELECT id, email, password, '4', '5', '6', '7', '8', '9' FROM Users--
```

S Der står qwert for kun at få brugerne ud, hvis man også vil have et produkt med kan man erstatte det med Apple Juice (1000ml). Dermed får man også det produkt med.

- **Explain how prepared statements prevent SQL injection**

SQL query og dataen bliver sendt til databasen separat, på en måde kan det ikke påvirke programmet. Statementet er prædefineret, og de variable dele af statementet bliver sat ind som variable og ikke en del af statementet. Det betyder man ikke kan tilføje til statementet udefra, hvilket forhindrer SQL injection.

- **Explain how to use placeholders in cases where prepared statements cannot do the job**

`SELECT * FROM clients WHERE clientID = $clientID`

Denne sql query gør brug af en placeholder der hedder clientID. Problemet med denne query er dog stadig at selvom der er en placeholder, så gælder det stadig, at så længe at en anden bruger har adgang mulighed for at tilføje sql kode til queryen, kan de lave injections.

`DECLARE clientID = 0 OR 1=1`

Ovenstående har en bruger givet et input der hedder “0 OR 1=1”, og selvom det er en placeholder, vil den endelige query se ud som følgende:

`SELECT * FROM clients WHERE clientID = $clientID`

Og siden at `$clientID = 0 OR 1=1` ender man med en query som følgende:

`SELECT * FROM clients WHERE clientID = 0 OR 1=1`

- **Explain how logging could be used to monitor injection attempts**

Ved at logge alt input data vil man kunne kigge efter sql kode fra det data der er blevet sat ind. Finder man sql kode har der nok været et injektion forsøg.