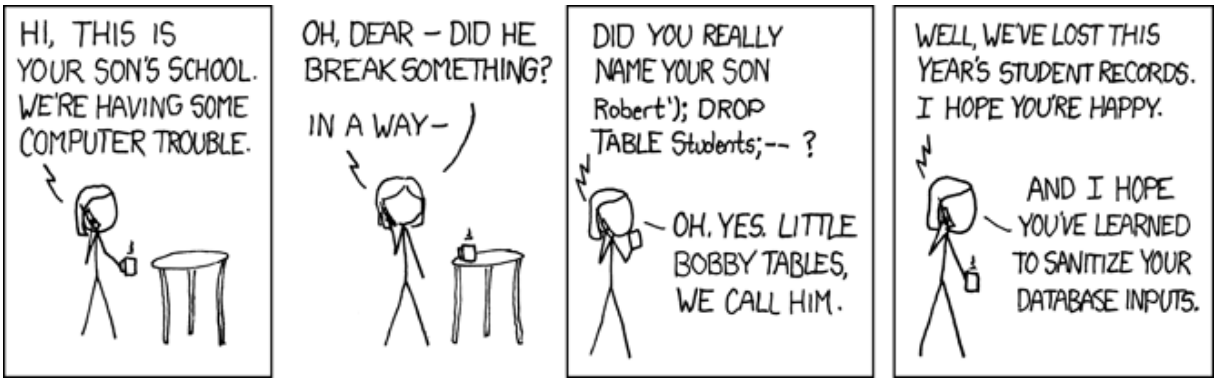
Opdracht 4: Security



(Bron: http://xkcd.com/327/)

1 Doel

In deze opdracht leer je over de werking van de meest voorkomende web applicatie vulnerability, SQL injections, d.m.v. het schrijven van software die gevoelig is voor SQL injections, gevolgd door het toepassen van database rollen, views en het gebruik van prepared statements om de database/applicatie te beveiligen.

2 Uitvoering

- Groepsgrootte: maximaal 2 studenten.
- De deadline: de eerst volgende ingeroosterde les.

Elk groepslid is verantwoordelijk voor alle gemaakte keuzes van het ingeleverde werk. Het cijfer wordt bepaald door de zwakste schakel binnen de groep. Je bent dus eigenlijk verplicht om het werk dat je hebt gemaakt uit te leggen aan je mede groepsgenoot, anders gaat het ten koste van je eigen groepscijfer.

3 Case

Een school heeft studenten in het systeem staan. Elke student heeft een unieke id, een naam, een wachtwoord, de klas (als een willekeurige string) en een vinkje dat aangeeft of de student wel/niet nog ingeschreven op de school is. Bij het invoeren van een correcte combinatie van de unieke id en wachtwoord, kan de student inloggen in het systeem.

De meest gebruikte functionaliteit van het systeem is:

1. studenten kunnen inloggen in het systeem, waarna alle gegevens van de student worden getoond
2. personen (iedereen) kan een klas invoeren, waarna alle alléén nog ingeschreven studenten uit die klas worden weergegeven (id, naam, klas, en wel/niet ingeschreven).

In het systeem gebeuren er rare dingen: 1) de studenten tabel verdwijnt regelmatig, 2) studenten lijken op een één of andere manier de gegevens van andere studenten te kunnen inzien en 3) er zwerft op Internet een lijst met alle studenten, ondanks dat deze informatie niet direct beschikbaar is. Jij (als student) vermoedt dat de problemen met SQL injecties te maken hebben. Om de school een handje te helpen, ga je demonstreren dat dergelijke problemen kunnen ontstaan door SQL injecties, en ga je vervolgens twee beveiligingsmaatregelen implementeren die de kans op dergelijke problemen minimaliseren.

4 Opdracht

4.1 Creëer de database en tabellen

Maak de database/tabel die voldoet aan de case.

4.2 Schrijf de applicatie

Schrijf een applicatie (bijv. een command-line applicatie) waarmee je de functionaliteit (zoals in de case beschreven is) mogelijk maakt.

Gebruik geen prepared-statements in je code, maar bouw zelf de query's (samen met de where clause waardes) op d.m.v. strings samen te voegen (string concatenation, in Java met de + operator). (Bijv: "SELECT * FROM studenten WHERE klas = '" + klas + "';"
i.p.v.
"SELECT * FROM studenten WHERE klas = ?;"')

Schrijf functionaliteit 1 op de volgende wijze: voer je query uit die checkt of de combinatie van de id en het wachtwoord een resultaat oplevert. Indien het een resultaat oplevert, toon dan de bijbehorende gegevens, anders geef aan dat de combinatie van het id en wachtwoord niet bestaat.

4.3 SQL Injectie 1: verwijder studenten

Voer een SQL injectie uit als gebruiker van de applicatie, waarmee je alle studenten verwijdert.

4.4 SQL Injectie 2: by-pass login

Voer een SQL injectie uit als gebruiker van de applicatie, waarmee je de gegevens van een andere student kan inzien (zonder daarbij het wachtwoord van te voren te weten).

4.5 SQL Injectie 3: alle gegevens

In functionaliteit 2 moet de klas worden ingevoerd, waarna alle alleen nog ingeschreven studenten uit die klas zichtbaar worden. Schrijf een SQL injectie waarmee alle studenten in het hele systeem worden weergegeven (dus ook uitgeschreven studenten en andere klassen).

4.6 Beveilig d.m.v. rollen i.c.m. views

Gebruik database rollen en views om de beschreven case meer te beveiligen. Bedenk zelf wat een goede opzet is.

4.7 Beveilig d.m.v. prepared statements

Beveilig de applicatie door prepared-statements te gebruiken.

Test of de SQL injecties nog mogelijk zijn na het gebruik van prepared-statements.

4.8 Extra

- Gebruik een website i.p.v. een cmdline app.
- Probeer SQLMap uit op een website van jezelf of van iemand van wie je toestemming hebt om SQLMap op uit te proberen, om vervolgens de gevoeligheid van die website voor SQL injecties mee te testen. (Let op dat je Python 2.6.x or 2.7.x gebruikt en niet een nieuwere versie.)

5 Inlevervorm

De uitwerkingen van de opdrachten dienen op papier (in een verslag) te worden genoteerd en mondeling te worden besproken en gedemonstreerd aan de docent.

Noteer op het verslag je voor- en achternaam, studentnummer, klas en datum van inleveren. Het verslag dient op de deadline dag te worden besproken met de docent. Verder dienen de simulaties op een computer gedemonstreerd te worden aan de docent.

6 Beoordelingscriteria

Onderwerp	Percentage van het opdrachtcijfer
SQL Injecties	28%
Beveiliging	28%
Motivatie in het verslag (Compleetheid van de beschrijving van de gemaakte keuzes. M.a.w. in hoeverre vallen de gemaakte keuzes af te leiden uit de tekstuele beschrijving/motivatie in het verslag.)	14%
Extra (alleen mogelijk indien alle andere opdrachtonderdelen afgerond zijn.)	30%