1. **Explica quin és l'objectiu d'aquesta pràctica.**

L'objectiu de la pràctica és prendre consciència que és molt fàcil que les dades de la nostra base de dades es vegin compromeses si no apliquem bones pràctiques (Prepared Statements).

2. **Escriu com queda la cadena $query amb les dades enviades (el passwd escrit al formulari eś: 'patata')**

```
SELECT * FROM access WHERE username='$user' AND password='patata';
```

3. **Explica que significa # en mysql i per què l'utilitza l'atacant.**

És un comentari que va fins al final de la línia, entenc que ho fa servir per tal que no s'executi cap més codi, i així no es corromp la consulta que fa. En concret, la query quedaria tal que

SELECT * FROM access WHERE username=''OR 1=1 ' and password='patata'
Com que volem eliminar aquesta cometa flotant, apliquem un comentari.

Un altre mètode seria INSERIR ' OR '1' = '1
i així no hauríem de posar un comentari.

4. **Explica per què aquesta pàgina és vulnerable a SQL Injection.**

Bàsicament el problema és que, en agafar dades directament del POST i afegirles a un SELECT, ens poden entrar altres consultes dins el POST, i fer consultes en comptes de donar-nos un usuari / password.

5. **Milloraria la seguretat usar POST en comptes de GET en aquest escenari? Com? Evitaria l'SQL Injection? Com?**

No, les dades s'envien de la mateixa manera. El POST no és invisible, es pot veure igual les dades, el POST MAI ens proporciona seguretat addicional. No és capaç d'evitar l'SQL Injection.

6. **Quin tipus de llibreries està usant el codi php? "Vendor Specific Database Extensions" o "Abstraction Layers". Influeix això amb la vulnerabilitat SQL Injection? Com?**

Està fent servir Vendor Specific Database Extensions (Per bases de dades MySQL,és igual si fas servir això o Abstraction Layers. Abstraction Layers és simplement això, un layer, però si a dins es fan males pràctiques i hi ha codi vulnerable a SQL Injection, el resultat és el mateix.

7. Explica com ho faries per tal que el codi PHP deixi de ser vulnerable a SQL Injection.

Primer, cambiaria el mysql per mysqli, que és més modern i suportat, que són una manera de fer consultes SQL segures. A bind_param, hi especifiquem ss, que vol dir que el primer paràmetre i el segon són strings.

```
<?php
   /* Connect to the Database */
   $dbLink = mysqli_connect("localhost", "username", "password");

   if (!$dbLink) {
      echo 'db link fail';
   }

   /* Select the database */
   mysqli_select_db($dbLink,"databaseName");

   /* Query and get the results */
   $user = $_GET["user"];
   $pass = $_GET["password"];

   $query = $mysqli->prepare("SELECT * FROM access WHERE username=? AND
password=?)
   $query->bind_param("ss", $user, $pass);
   $result = mysqli_query($query);

   /* Check results */
   $row = mysqli_fetch_array($result, MYSQL_ASSOC));
   if (!$row){
      die("Error authenticating");
   }
```

8. Un cop llegit el codi php que processa les dades, com ompliries el formulari per a crear el teu propi usuari a la base de dades?

Si el codi admetés multiquery, seria tan fàcil com introduir això al camp password mateix: hola' OR 1=1; INSERT INTO acess(`username`,`password`) VALUES (‘alex’,’1234’); #