This lab from PortSwigger demonstrates an **SQL injection vulnerability** in the WHERE clause of an SQL query. The goal is to exploit the vulnerability to retrieve hidden data from the application's database.

**Steps to Solve the Lab**

1. **Understanding the Scenario**:
   * The web application filters and displays items based on a search query or ID, often using a URL parameter or input field.
   * The SQL query behind the scenes may look like this:

SELECT \* FROM products WHERE category = 'user\_input';

* + my objective is to modify the query in such a way that additional or hidden data is retrieved.

1. **Testing for SQL Injection**:
   * Navigate to the lab's product page or search field.
   * Begin by entering a simple SQL injection payload such as:

' OR 1=1—

* + This payload will modify the query to:

SELECT \* FROM products WHERE category = '' OR 1=1--';

* + The OR 1=1 condition will always be true, causing the query to return all products, including hidden ones.

1. **Injecting Payload**:
   * Use the payload category=' OR 1=1-- in the URL or input field.
   * Example if the URL looks like this:

<https://example.com/products?category=toys>

* + Modify it to:

https://example.com/products?category=' OR 1=1--

* + When executed, the application will return all records, including hidden or restricted data.

1. **Validate and Submit**:
   * If the payload is successful, you will see hidden data such as other product categories or items that were previously not visible.
   * After retrieving the hidden data, follow the lab instructions to complete the task.

#### How the Attack Works:

* **' OR 1=1--**: The OR 1=1 condition is always true, bypassing the intended filter of the query. The -- is a comment operator in SQL, which tells the database to ignore the rest of the query, preventing errors.

#### Preventing SQL Injection:

To prevent SQL injection, always use **parameterized queries** or **prepared statements**. These techniques ensure that user input is treated as data rather than executable code. For example, using parameterized queries in code:

SELECT \* FROM products WHERE category = ?;