The issue at hand is SQL Injection on the product page, as when a user enters something in the search bar, they could potentially inject SQL code and manipulate the database as they please.

In the file app/php/product.php, there was not proper validation of MySQL queries. To fix this issue, in app\_sec/php/product.php, we switched to using prepared statements, eliminating SQL Injections.

**Demonstration of the corrected vulnerability:**

Here we have the products displayed on the page that were initially retrieved from the database:Uma imagem com texto, captura de ecrã, software, design

Descrição gerada automaticamente

An attacker can inject an SQL command into the search bar to change the price of a product named "Product 62" (the first product listed) to "'; UPDATE products SET price = '0.0' WHERE name = 'Product 62';--".Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

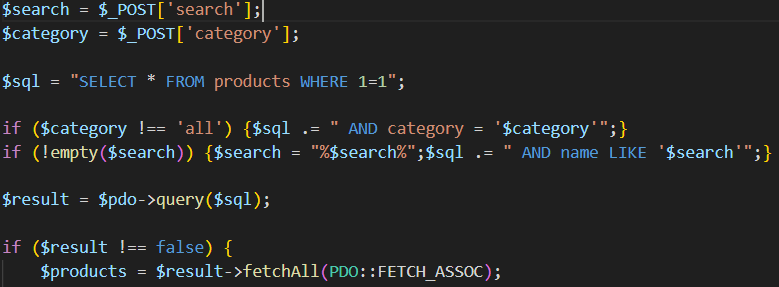
Here we can see that with the SQL command, the price of "Product 62" was changed.

As mentioned earlier, to fix this issue, we have used prepared statements to ensure proper data handling. Thus, in the secure application, when the attacker executes the same SQL command as before, nothing happens because the input in the text box clearly does not match the name of a product in the application.Uma imagem com texto, captura de ecrã, Tipo de letra, design

Descrição gerada automaticamente

**Comparison of the corrected code:**

Insecure code:



Secure code:  
