OWASP top ten – Web Application Security

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications.

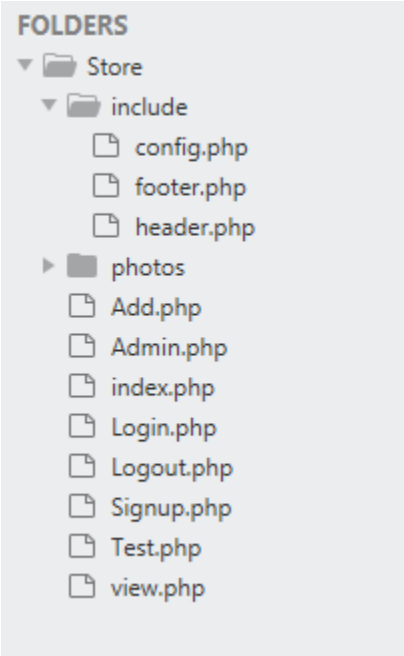
We designed a simple vulnerable web application that includes 3 vulnerabilities of OWASP top ten, we used the following programming, scripting, designing and query languages:

1-HTML(FRONT-END language)
2-CSS(STYLING language)
3-Bootstrap(FRONT-END library)
4-JavaScript(FRONT-END language)
5-PHP(BACK-END language)
6-SQL(QUERY language)

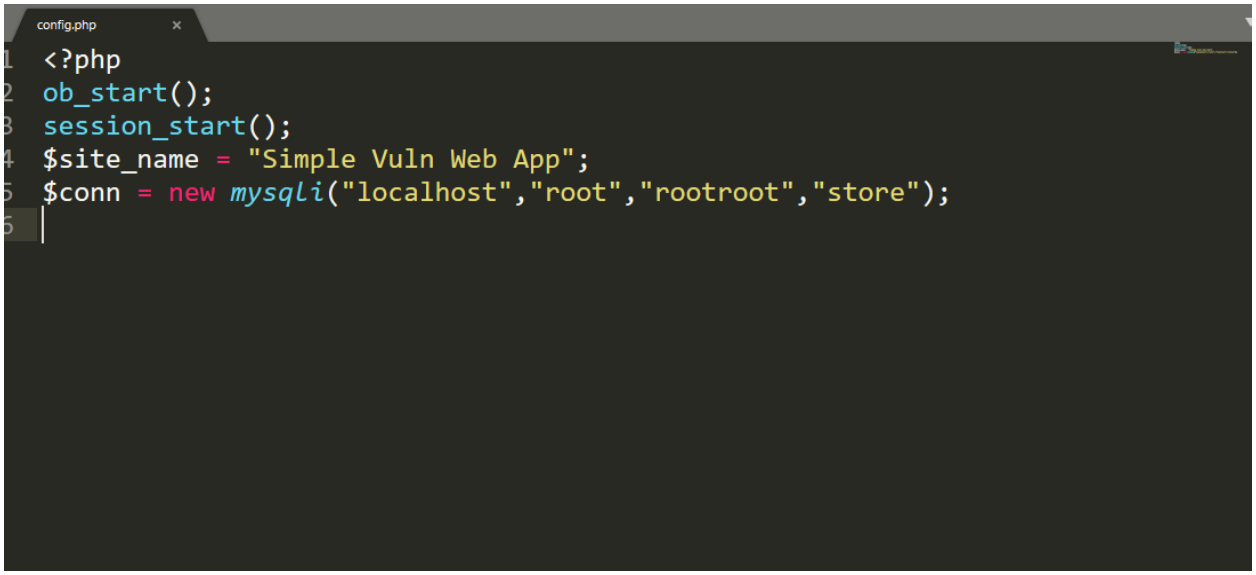
The Web app includes following vulnerabilities:

1-SQLI (SQL Injection)
2-XSS (CROSS SITE SCRIPTING)
3-Broken Access Control

And it has these files:



The config file contains the site settings and the database settings



The footer and the header have just the design of them

```php
<?php
$sql = "SELECT * FROM products";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
?>

      <div class="col">
        <div class="card">

                  <div style="  background-image: url(<?php echo 'photos/'.$row['photo']?>);
  background-color: #cccccc;
  height: 200px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
">
          </div>
          <div class="card-body">
            <h5 class="card-title"><?php echo $row['title']?></h5>
            <a class="btn btn-success" href="view.php?id=<?php echo $row['id']?>">View</a>
          </div>
        </div>
      </div>


<?php
```

Tab Size: 4 PHP

The index file does include the header and the footer and get the products from the Database using **SELECT * FROM products**

```php
<?php
include 'include/config.php';
if(isset($_POST['login'])){
    if(!empty($_POST['username']) and !empty($_POST['password'])){
        $username = $_POST['username'];
        $password = $_POST['password'];
        $sql = "SELECT * FROM users where username = '$username' and password = '$password'
            LIMIT 1 ";
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            while($row = $result->fetch_assoc()) {
                $_SESSION['id'] = $row['id'];
                $_SESSION['login'] = True;
            }
        }else{
            echo('Incorrect');
        }


    }else{
        echo('Please submit all the information');
    }
}
?>

<!DOCTYPE html>
<html>
<head>
    <title><?php echo $site_name;?></title>
```

Tab Size: 4 PHP

The login page does do the **SELECT query**, and if the query is true will create a session with the id of the user

```php
<?php
include 'include/config.php';
if(isset($_POST['signup'])){
    if(!empty($_POST['username']) and !empty($_POST['name']) and !empty($_POST['password'])){
        $username = $_POST['username'];
        $name = $_POST['name'];
        $password = $_POST['password'];
        $sql = "INSERT INTO users (username,name,password)values('$username','$name','$
            password')";
        if ($conn->query($sql) === TRUE) {
          echo "Congratulations, you are one of us now";
        } else {
          echo "Error: " . $sql . "<br>" . $conn->error;
        }
    }else{
        echo('Please submit all the information');
    }
}



?>
<!DOCTYPE html>
<html>
<head>
    <title><?php echo $site_name;?></title>
</head>
<body>
<?php include 'include/header.php';?>
```

The signup does **insert** the data with **insert query statement**

config.php | index.php | Login.php | Signup.php | view.php

```php
<?php
$sql = "SELECT * FROM products WHERE id='$id'";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
?>

      <div class="col container" >
        <div class="card">

                  <div style="  background-image: url(<?php echo 'photos/'.$row['photo']?>);
  background-color: #cccccc;
  height: 200px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
">
          </div>
          <div class="card-body">
            <h5 class="card-title"><?php echo $row['title']?></h5>
          </div>
        </div>
      </div>


<?php
    }
```

The view page does get the product using the id **GET METHOD**

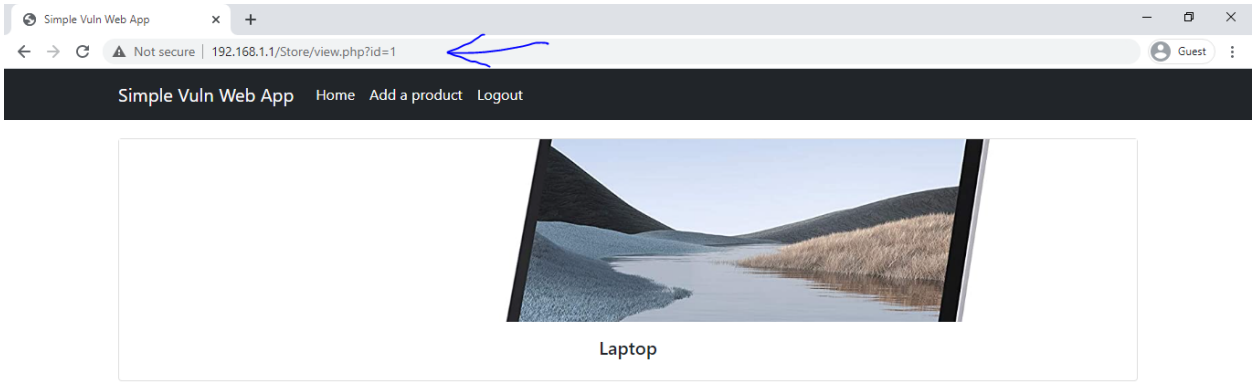
```php
<?php
include 'include/config.php';
if (!isset($_SESSION['login'])){
    header("Location: index.php");
    die();
}

if(isset($_POST['add'])){
    var_dump($_POST);
    $title = $_POST['title'];
    $userId =  $_SESSION['id'];
    $image_name = $_FILES['photo']['name'];
    $image_tmp =  $_FILES['photo']['tmp_name'];
    $UP = move_uploaded_file($image_tmp, 'photos/'. $image_name);
    $sql = "INSERT INTO products (id,user,photo,title)VALUES('','$userId','$image_name','$
        title')";
    if ($conn->query($sql) === TRUE) {
        echo "Congratulations, your product was added";
    }

}


?>
<!DOCTYPE html>
<html>
<head>
    <title><?php echo $site_name;?></title>
</head>
```

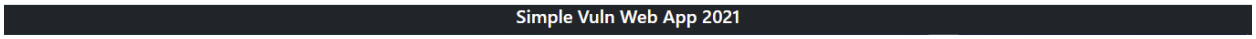
The add page does **insert new products** from any user

```php
<?php
$sql = "SELECT * FROM products";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
?>

      <div class="col">
        <div class="card">

                  <div style="  background-image: url(<?php echo 'photos/'.$row['photo']?>);
  background-color: #cccccc;
  height: 200px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
">
          </div>
          <div class="card-body">
            <h5 class="card-title"><?php echo $row['title']?></h5>
            <a class="btn btn-success" href="view.php?id=<?php echo $row['id']?>">View</a>
            <a class="btn btn-danger" href="delete.php?id=<?php echo $row['id']?>">Delete</a>
          </div>
        </div>
      </div>
```

And the admin page does get data and delete any product we want from the database

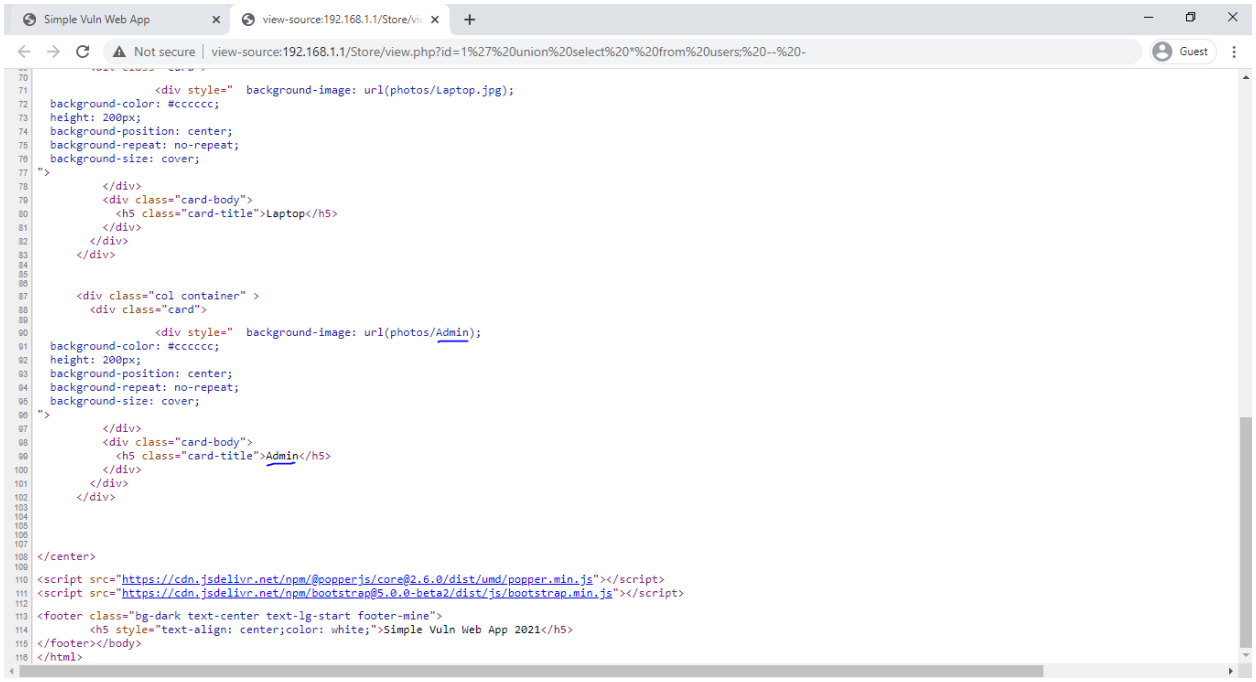
Now we are going to create an account to find the vulnerabilities and exploit them





We found that the view button in the home page does go to view.php?id=and the id of the product
http://192.168.1.1/Store/view.php?id=1
Now we will try to add ‘ to see if this page is vulnerable to sqli or not
It does show us a sql error, we will try to get the users:

http://192.168.1.1/Store/view.php?id=1’ union select * from users; -- -



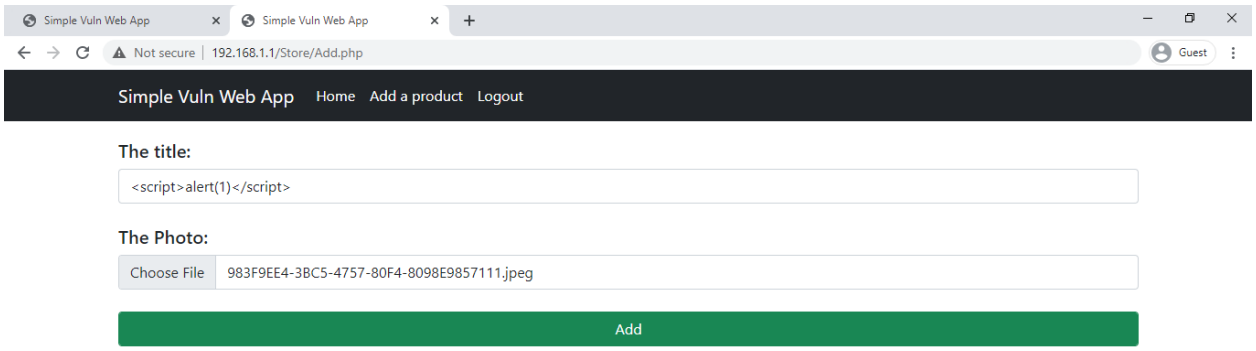
Now we do have the user and password of the admin

This vulnerability calls SQLI where we can change the query the bug is on the view file

```php
<?php include 'include/header.php';?>
<div style="padding-top:20px;"></div>



<?php
$sql = "SELECT * FROM products WHERE id='$id'";//here we can do sqli ex id = '1'or 1=1-- -'
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
?>

      <div class="col container" >
        <div class="card">

                    <div style="  background-image: url(<?php echo 'photos/'.$row['photo']?>);
  background-color: #cccccc;
  height: 200px;
  background-position: center;
  background-repeat: no-repeat;
  background-size: cover;
">
          </div>
          <div class="card-body">
            <h5 class="card-title"><?php echo $row['title']?></h5>
          </div>
        </div>
      </div>

```

The best way to fix it is to use PDO with prepared statements

When we were trying to add a new product we found that we can inject a javascript code in the title of the product



Now when we do visit the home page this alert come to us, because the javascript code is on the page, this vulnerability calls XSS, we can add whatever javascript code we want, we can get the cookie of the users too when the do visit the home page

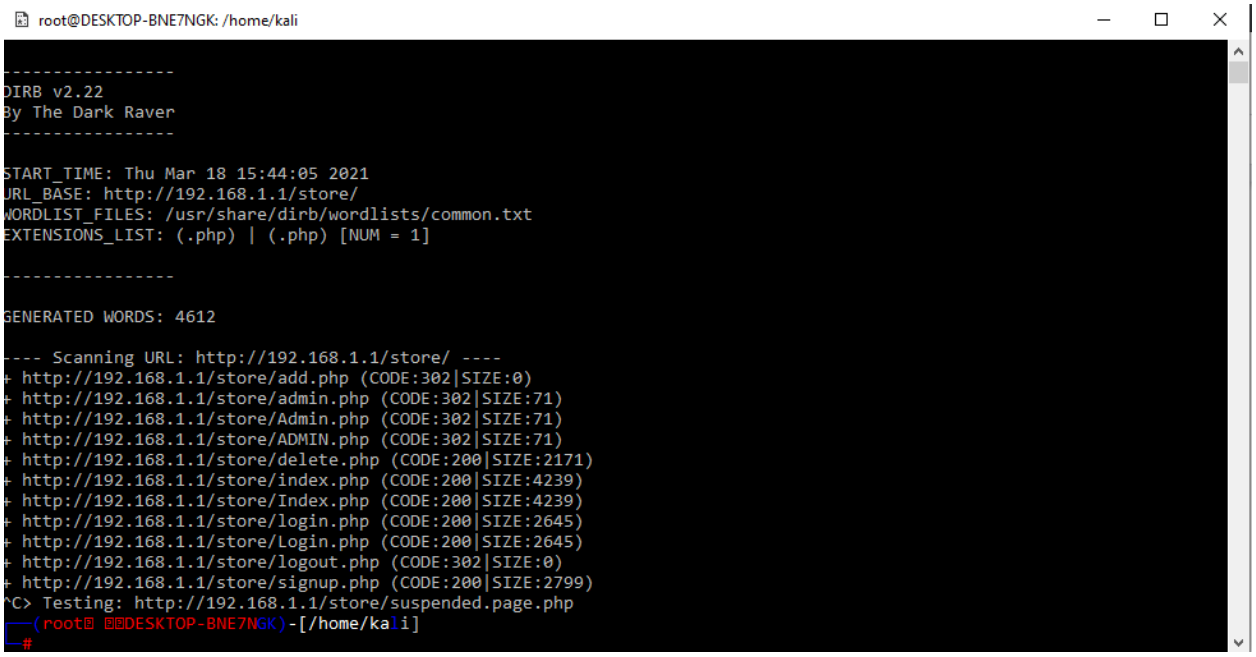
The locate of the bug is here

```php
<?php
include 'include/config.php';
if (!isset($_SESSION['login'])){
    header("Location: index.php");
    die();
}

if(isset($_POST['add'])){
    var_dump($_POST);
    $title = $_POST['title'];//the user can add a javscript code here
    $userId =  $_SESSION['id'];
    $image_name = $_FILES['photo']['name'];
    $image_tmp =  $_FILES['photo']['tmp_name'];
    $UP = move_uploaded_file($image_tmp, 'photos/'. $image_name);
    $sql = "INSERT INTO products (id,user,photo,title)VALUES('','$userId','$image_name','$
        title')";
    if ($conn->query($sql) === TRUE) {
        echo "Congratulations, your product was added";
    }

}


?>
<!DOCTYPE html>
<html>
<head>
    <title><?php echo $site_name;?></title>
</head>
```

The best way to fix it is to add the htmlentities() function, so the value will ignore the html codes

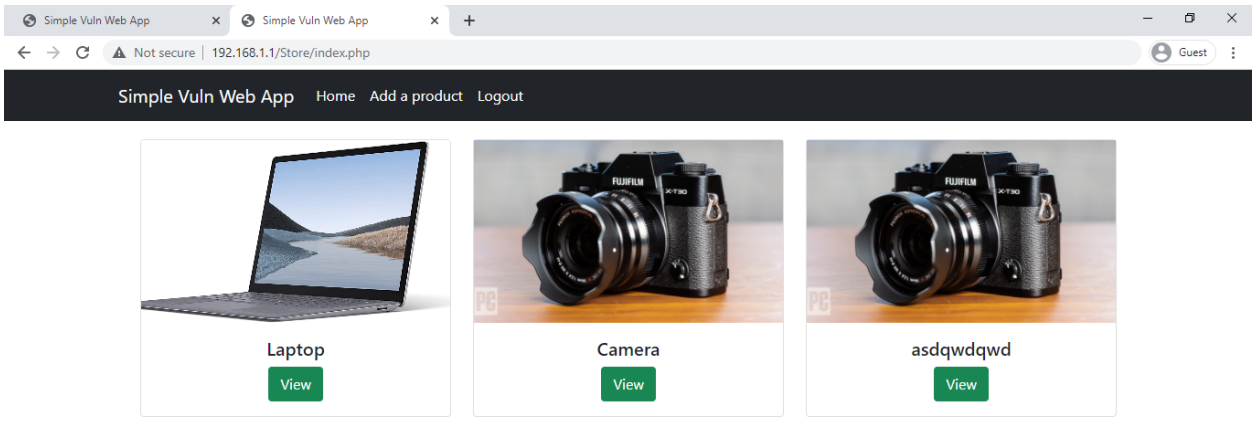
Now we are going to do a brute force attack to find the files or the folders we can see using dirb tool

The command is
dirb http://192.168.1.1/store -X .php
The -X is to select the extension
After some minutes we found these pages



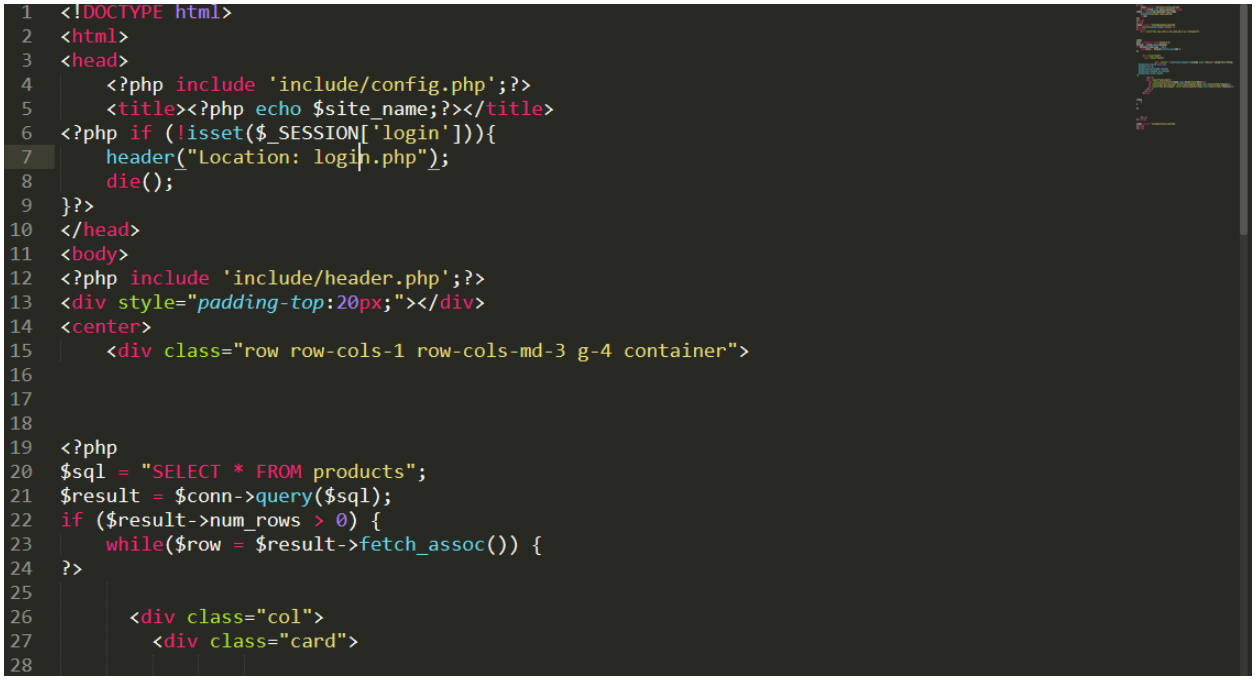
There’s a page called admin.php

And after visiting it we found that we can access it with a normal user



This vulnerability calls Broken Access Control where we can broke the access control just doing brute force in the directories and files

The bug is in the admin.php page



```php
<!DOCTYPE html>
<html>
<head>
    <?php include 'include/config.php';?>
    <title><?php echo $site_name;?></title>
<?php if (!isset($_SESSION['login'])){
    header("Location: login.php");
    die();
}?>
</head>
<body>
<?php include 'include/header.php';?>
<div style="padding-top:20px;"></div>
<center>
    <div class="row row-cols-1 row-cols-md-3 g-4 container">



<?php
$sql = "SELECT * FROM products";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
?>

      <div class="col">
        <div class="card">

```

It does ask if there's a login session or not, if so it will return the page and give the access to the users

The best way to fix this bug is to check if the users is an admin or not

References:

https://www.php.net/

https://getbootstrap.com/

https://www.w3schools.com/

https://owasp.org/www-project-top-ten/

https://tools.kali.org/web-applications/dirb