SQL Injection Vulnerability Report

Affected Product

Attribute	Details
Product Name	Online Shopping Portal Project
Vendor	PHPGurukul
Version	v2.1
Affected File	Online Shopping Portal project-V2.0\shopping\product-details.php
Affected Parameter	name
Method	POST
Vulnerability Type	Time-Based Blind SQL Injection

Official Website

https://phpgurukul.com/shopping-portal-free-download/

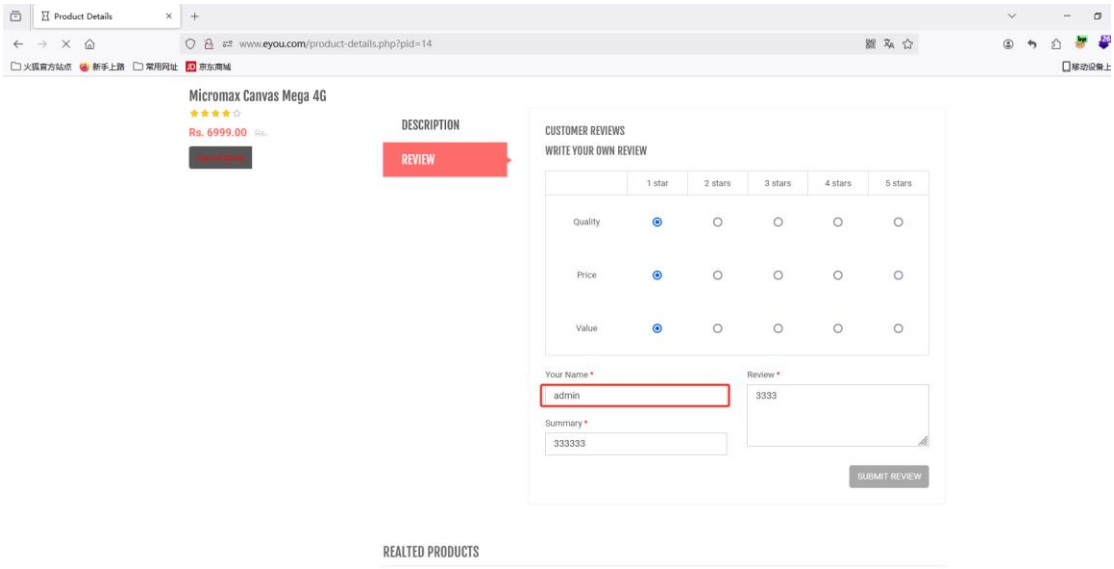
Vulnerability Overview

A SQL Injection vulnerability exists in the **name** parameter of the **Online Shopping Portal Project v2.1**, allowing remote attackers to execute arbitrary SQL commands. By injecting time-delay payloads, attackers can determine the presence of a SQL Injection flaw by observing server response delays.

Steps to Reproduce

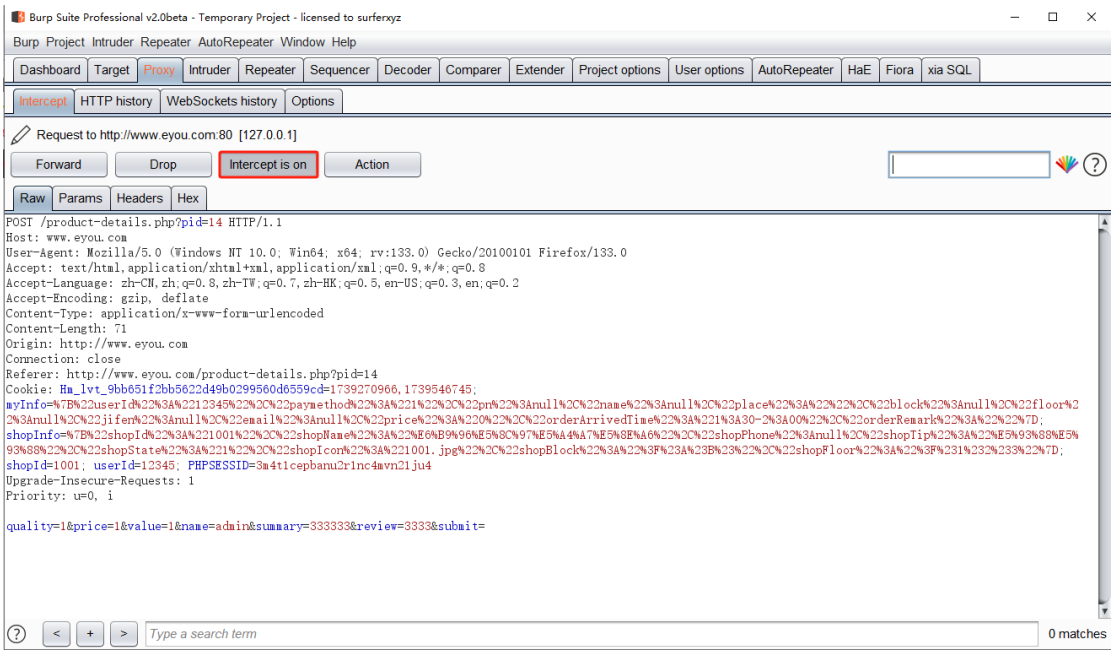
1. **Access the Vulnerable URL:**

   http://www.eyou.com/ product-details.php?pid=14



2. **Intercept the Request:**

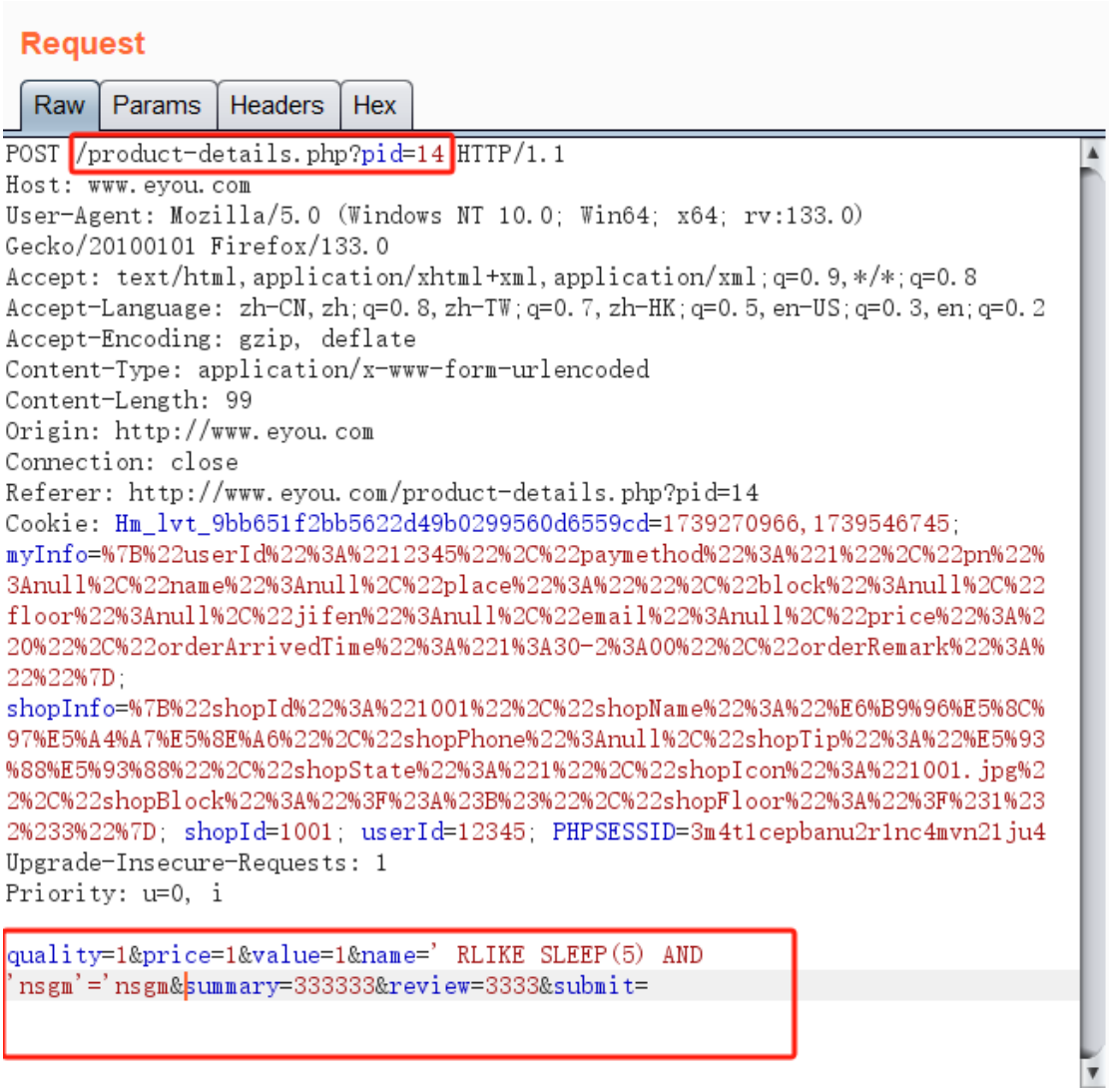
Enable Burp Suite and set up the browser to route traffic through it.



3. **Modify the Parameter:**

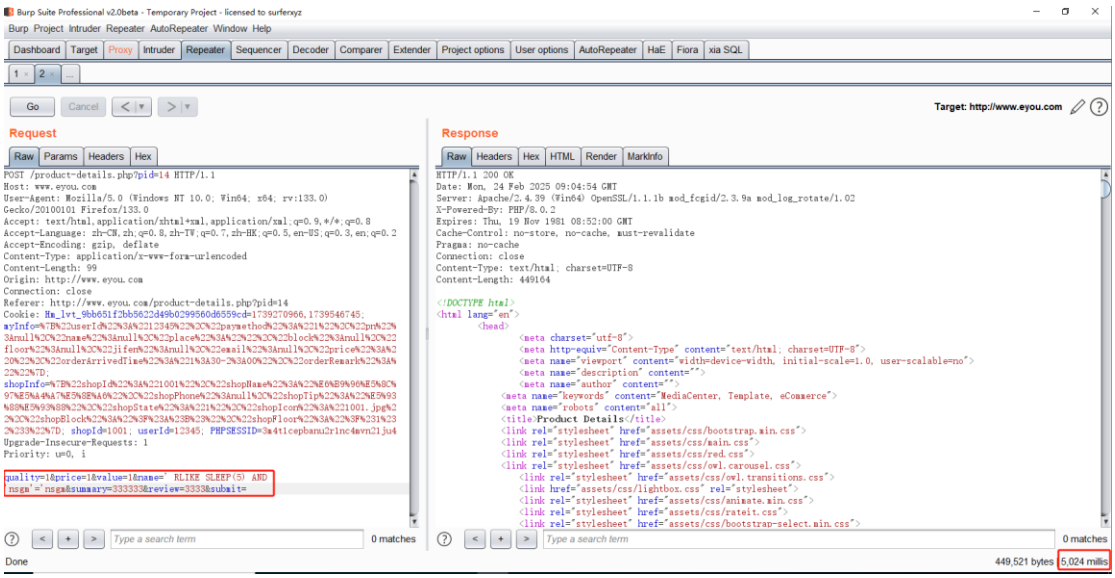
Send the request to Burp Suite Repeater and modify the name parameter with the following payload:

' RLIKE SLEEP(5) AND 'nsgm'='nsgm

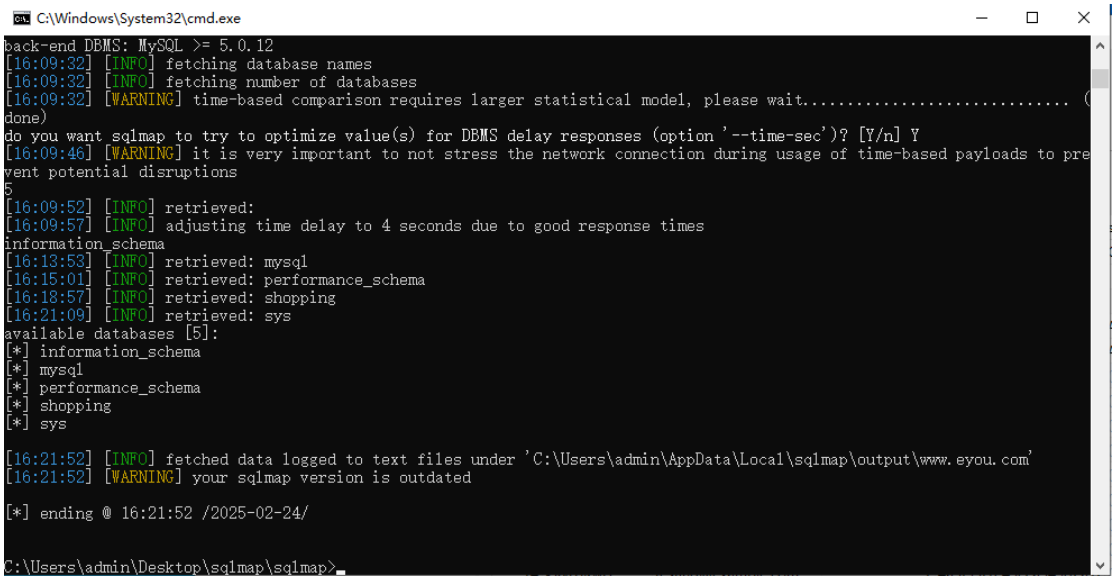


4.Send the Modified Request:

- Forward the modified request in Burp Suite Repeater.
- Observe the delay in the response time.
- The server will delay its response by 5 seconds, confirming successful execution of the SLEEP() function, indicating a **time-based SQL injection vulnerability**.

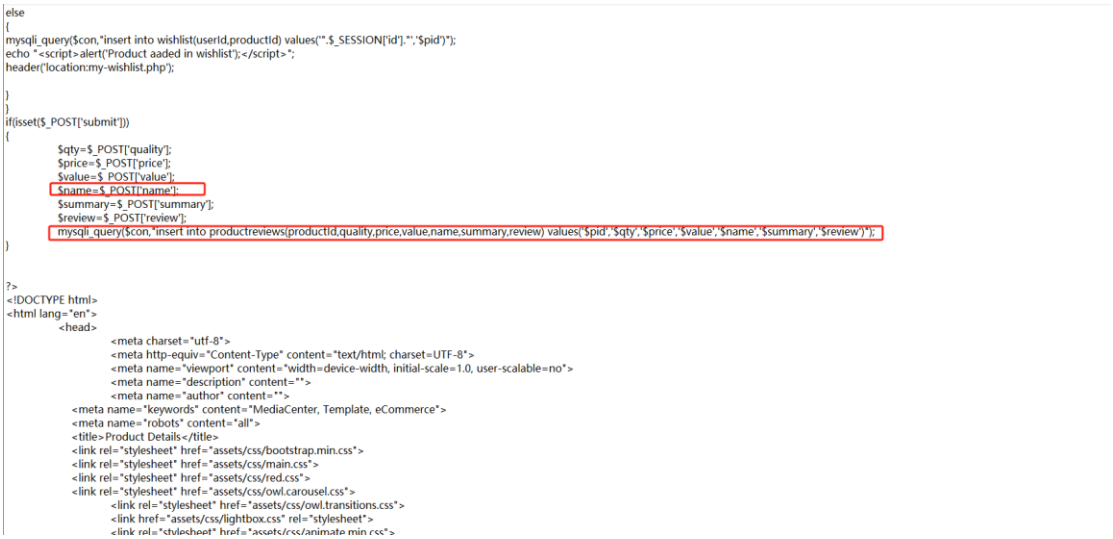


SQLmap:



Code

In the Online Shopping Portal project-V2.0\shopping\ product-details.php page, the **name** parameter is not verified and is directly inserted into the database for execution



```
else
{
mysqli_query($con,"insert into wishlist(userId,productId) values('".$_SESSION['id']."','$pid')");
echo "<script>alert('Product aaded in wishlist');</script>";
header('location:my-wishlist.php');

}
}
if(isset($_POST['submit']))
{
        $qty=$_POST['quality'];
        $price=$_POST['price'];
        $value=$_POST['value'];
        $name=$_POST['name'];
        $summary=$_POST['summary'];
        $review=$_POST['review'];
        mysqli_query($con,"insert into productreviews(productId,quality,price,value,name,summary,review) values('$pid','$qty','$price','$value','$name','$summary','$review')");
}


?>
<!DOCTYPE html>
<html lang="en">
        <head>
                <meta charset="utf-8">
                <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
                <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
                <meta name="description" content="">
                <meta name="author" content="">
            <meta name="keywords" content="MediaCenter, Template, eCommerce">
            <meta name="robots" content="all">
            <title>Product Details</title>
            <link rel="stylesheet" href="assets/css/bootstrap.min.css">
            <link rel="stylesheet" href="assets/css/main.css">
            <link rel="stylesheet" href="assets/css/red.css">
            <link rel="stylesheet" href="assets/css/owl.carousel.css">
                <link rel="stylesheet" href="assets/css/owl.transitions.css">
                <link href="assets/css/lightbox.css" rel="stylesheet">
                <link rel="stylesheet" href="assets/css/animate.min.css">
```

Impact

- **Data Theft:** Unauthorized access to sensitive user or system data.
- **Data Manipulation:** Modification or deletion of database records.
- **Credential Exposure:** Extraction of usernames, passwords, or authentication details.
- **Server Compromise:** Potential exploitation of underlying server systems.
- **Reconnaissance:** Enumeration of database structures (tables, columns, schemas).
- **Financial Loss:** Downtime and potential monetary losses.
- **Loss of Reputation:** User trust degradation due to service disruption or data breaches.

Recommended Mitigations

- **Use Prepared Statements (Parameterized Queries).**
- **Sanitize User Inputs:** Validate and filter all incoming data.
- **Implement Web Application Firewall (WAF).**
- **Use the Principle of Least Privilege (PoLP) for database users.**
- **Regularly Update and Patch the Application.**
- **Monitor Logs for Suspicious Activities.**

For detailed guidelines, refer to:
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html