



PROJECT REPORT

Cyber Security (23CP310P)

Faculty
Dr. Aashka Raval

Teammates

@Barbie Sharma – 21BCP315

@Jainil Prajapati – 21BCP448D

@Jatin Prajapati – 21BCP452D

Experiment – 1

Aim: SQL Injection on a website (basic SQL attack).

- **What we had done for in this attack?**

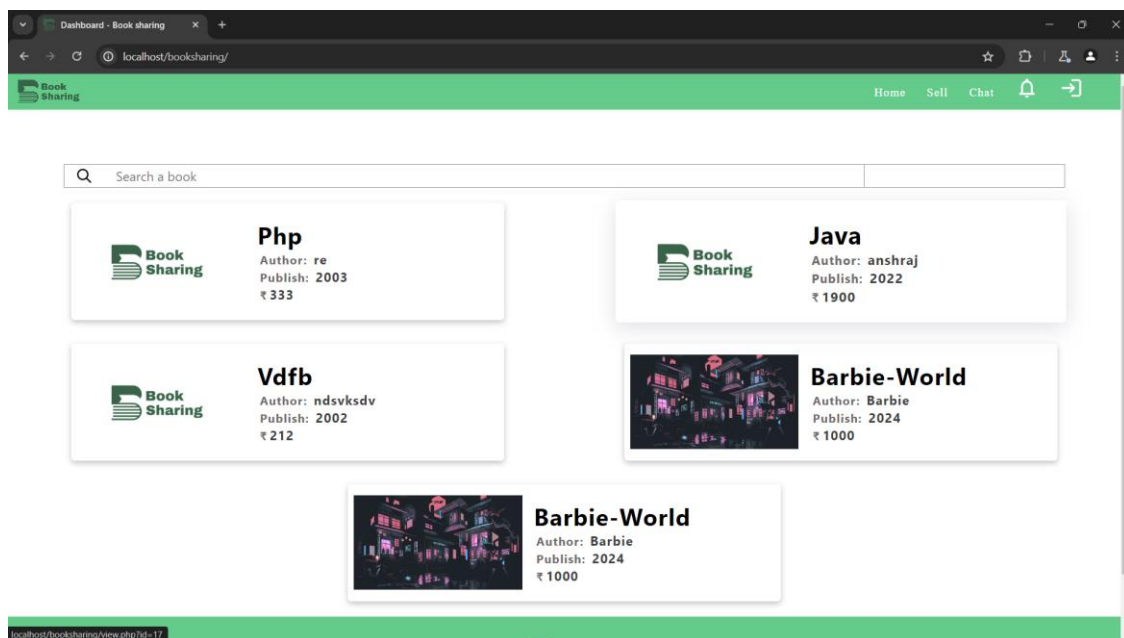
→ In this attack we had implemented the most basic attack of SQL Injection. We had a plan to do UNION Attack on the site but somehow that is not happening in our project. In that UNION Attack we tried to get all information from user table.

So, Finally we have come to conclusion to attack with most basic to demonstrate to the attack.

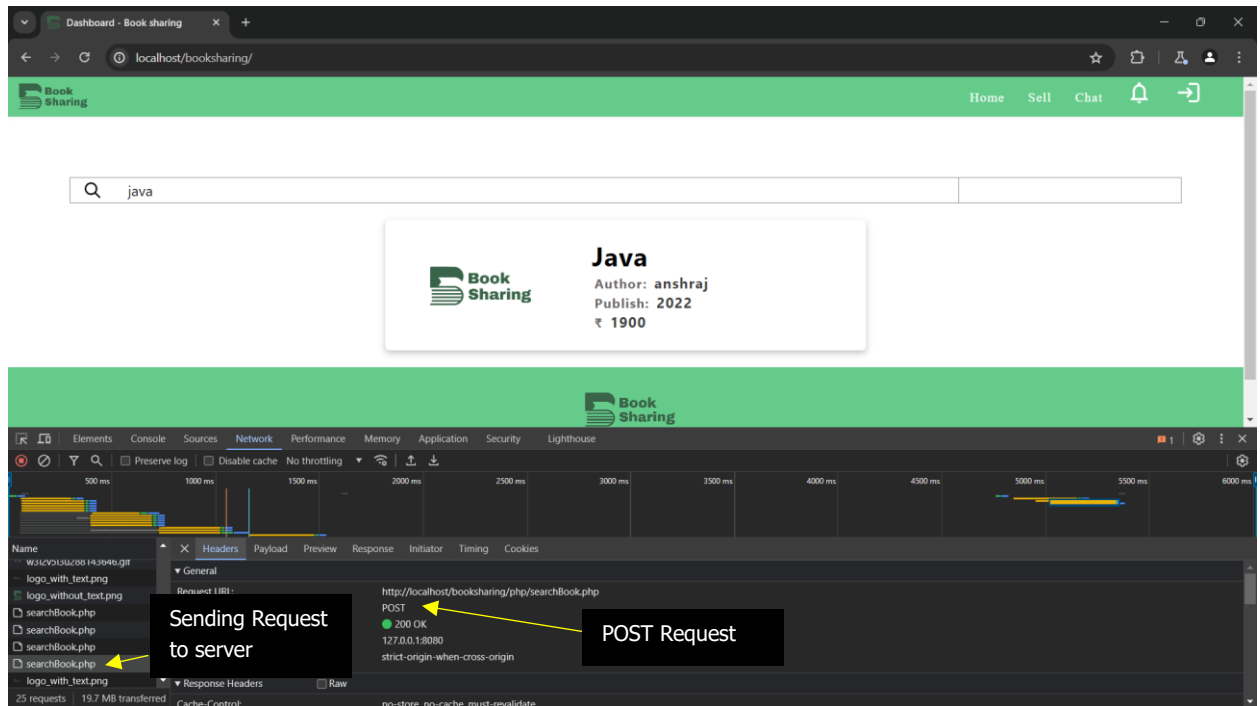
- **SQL Attack**

1) We have a website for selling and buying books.

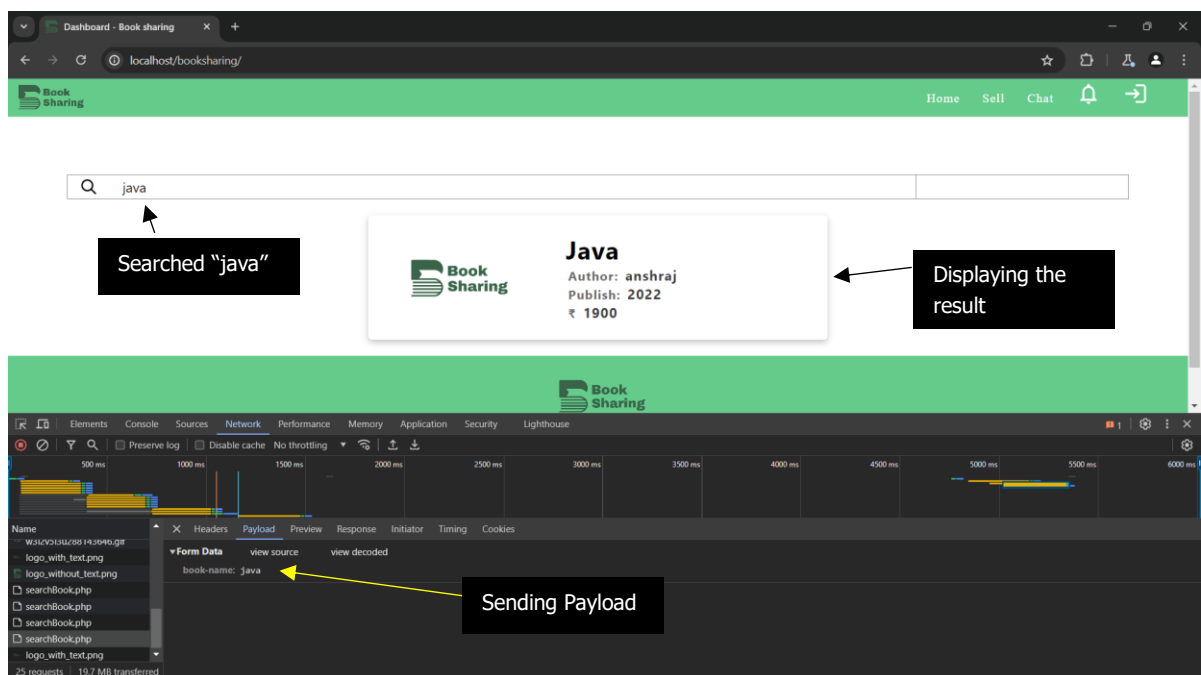
2) There were a Search Bar to search the book.



3) So, We found out that on the type it has sending request to the database for the fetching books.

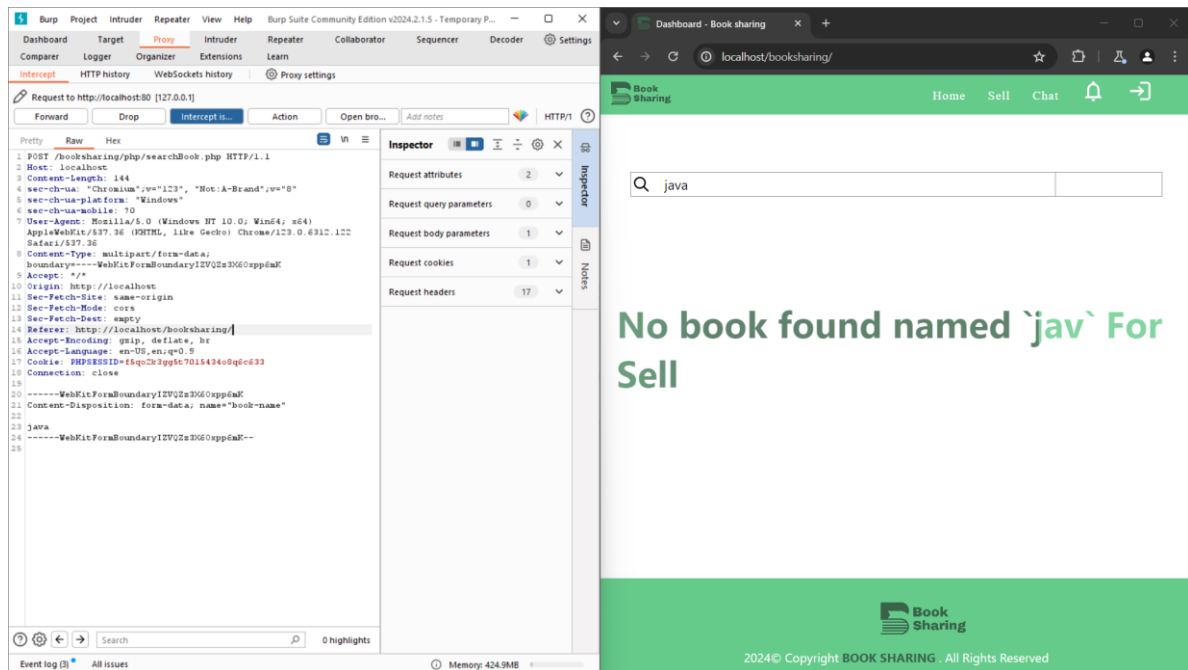


4) It is sending POST request we the payload of "book-name".

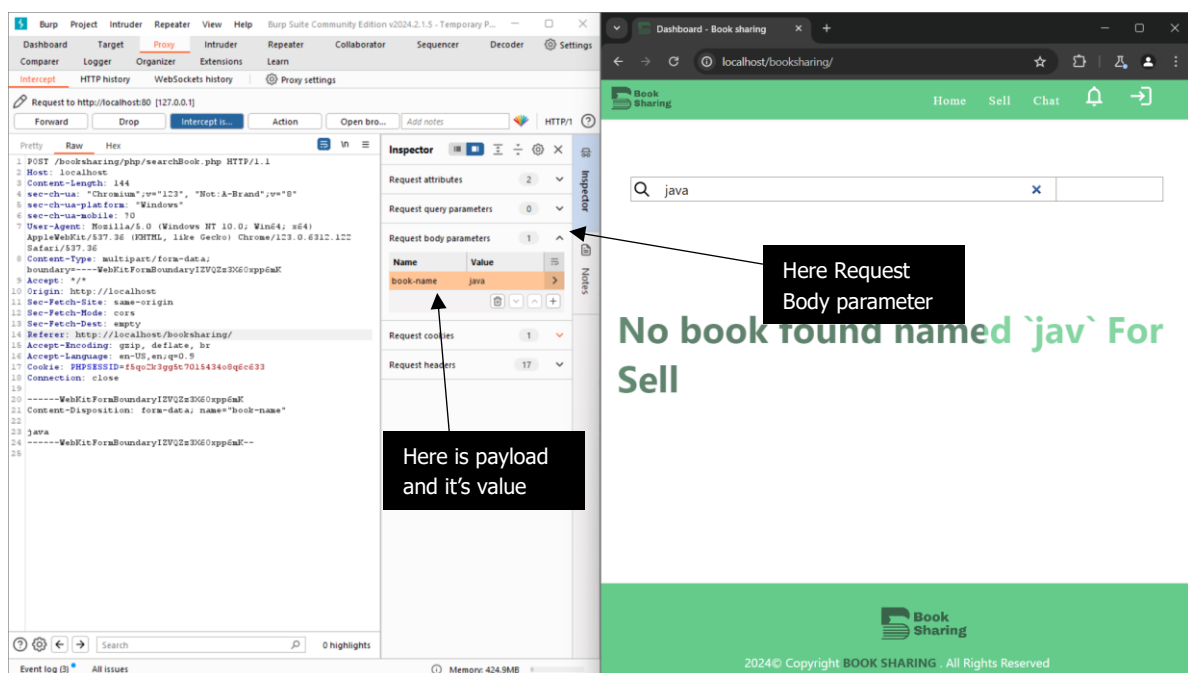


5)

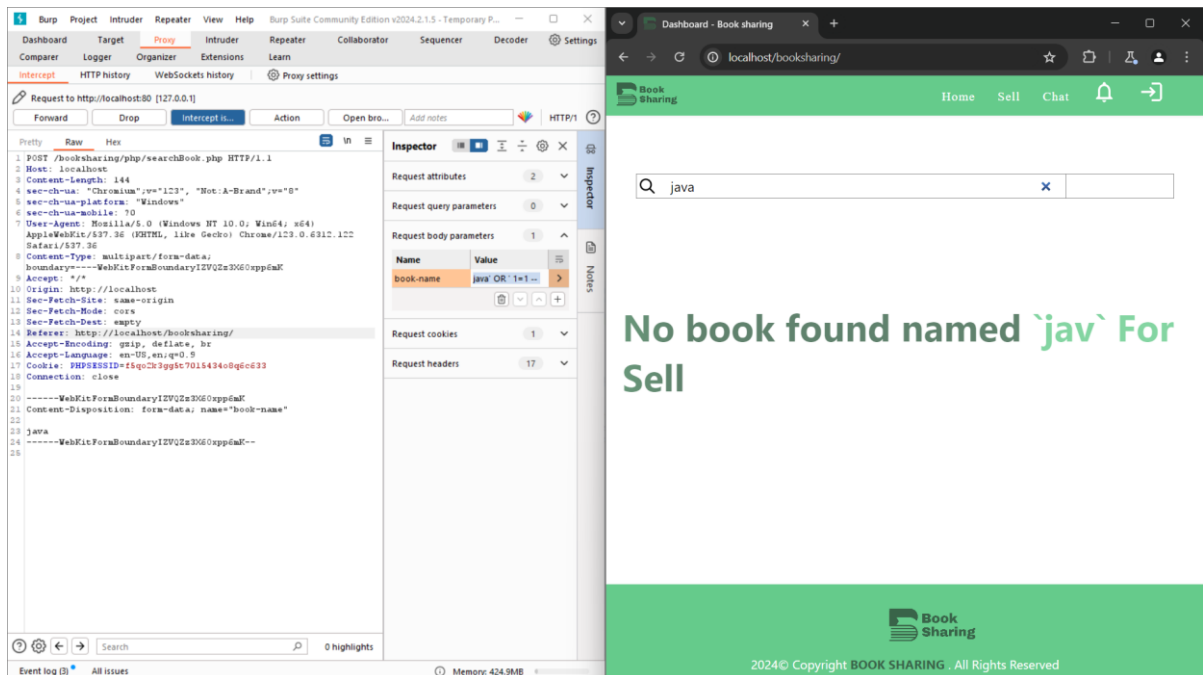
6) Connect this with Burp Suite and in proxy turn on proxy and do the same request.



7) Expand the request body parameters there you can find the payload "book-name": [some value...].

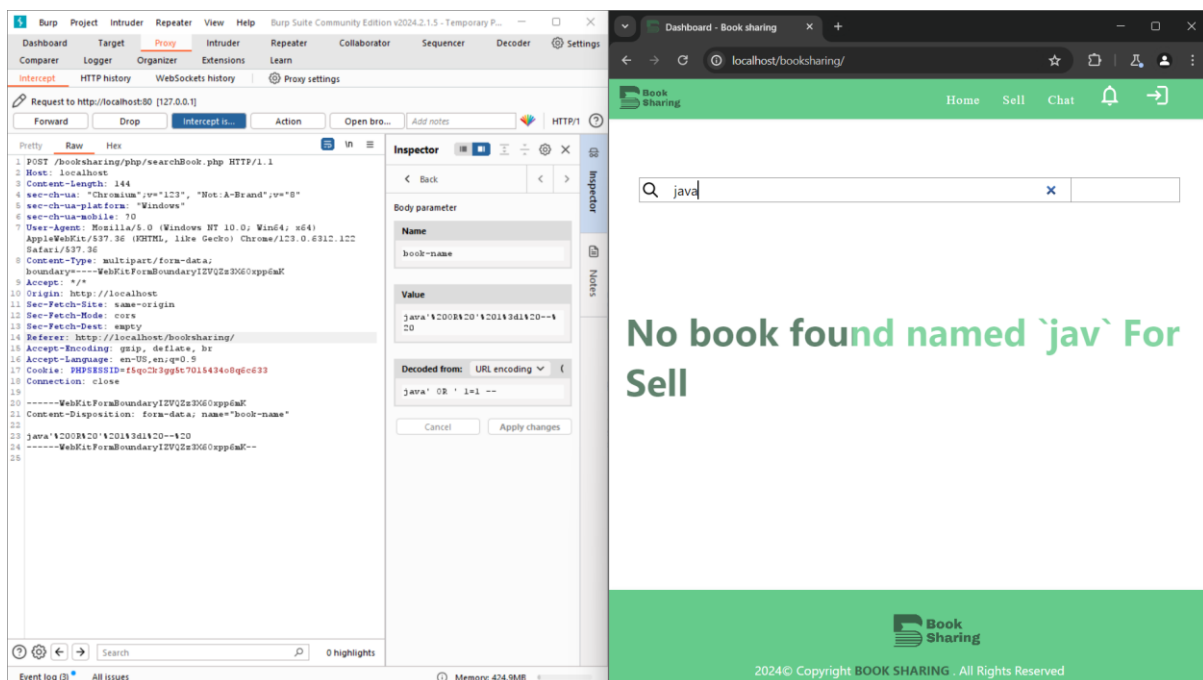


8) Append the `` ' OR ' 1=1 -- `` after the the value of that parameter. (in our case it is java so it will be `java' OR ' 1=1 --`)



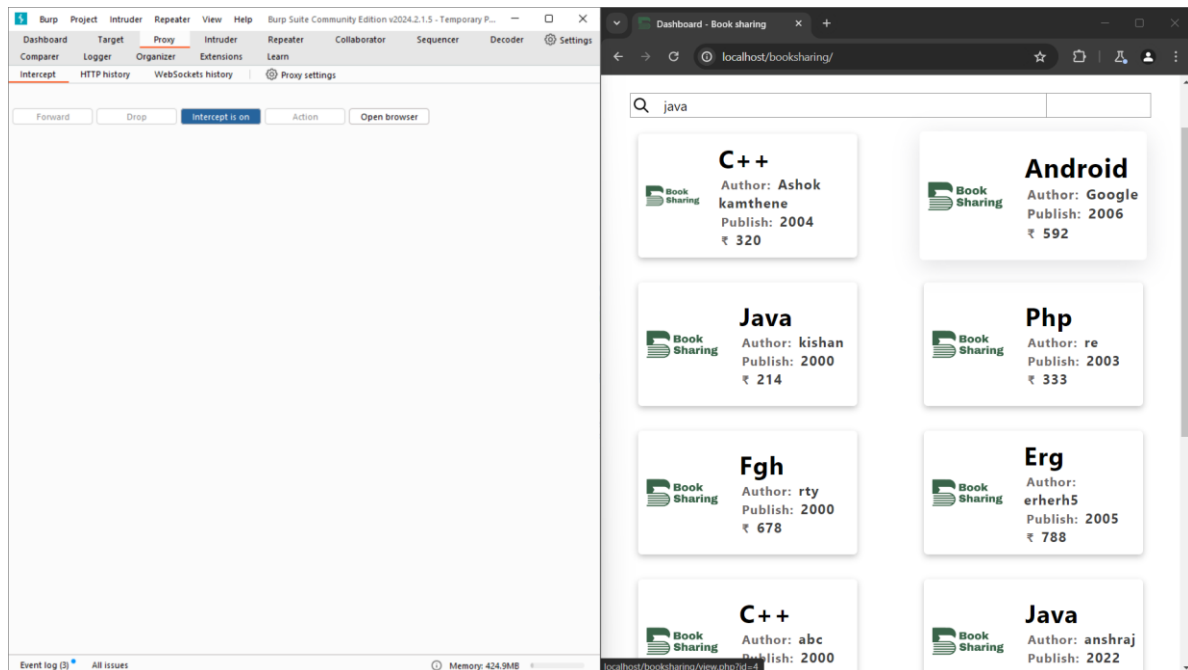
The screenshot shows the Burp Suite interface on the left and a web browser on the right. In Burp Suite, the 'Intercept' tab is active, showing a request to `http://localhost:80 [127.0.0.1]`. The 'Raw' tab displays the request body, which is a multipart/form-data request. The 'Inspector' tab shows the request parameters, with the 'book-name' parameter highlighted. The value of 'book-name' is `java' OR ' 1=1 --`. The web browser on the right shows the 'Dashboard - Book sharing' page with a search bar containing 'java'. Below the search bar, a message reads: 'No book found named `jav` For Sell'.

9) To edit just double click on the value and after editing click on the arrow right side of the value.



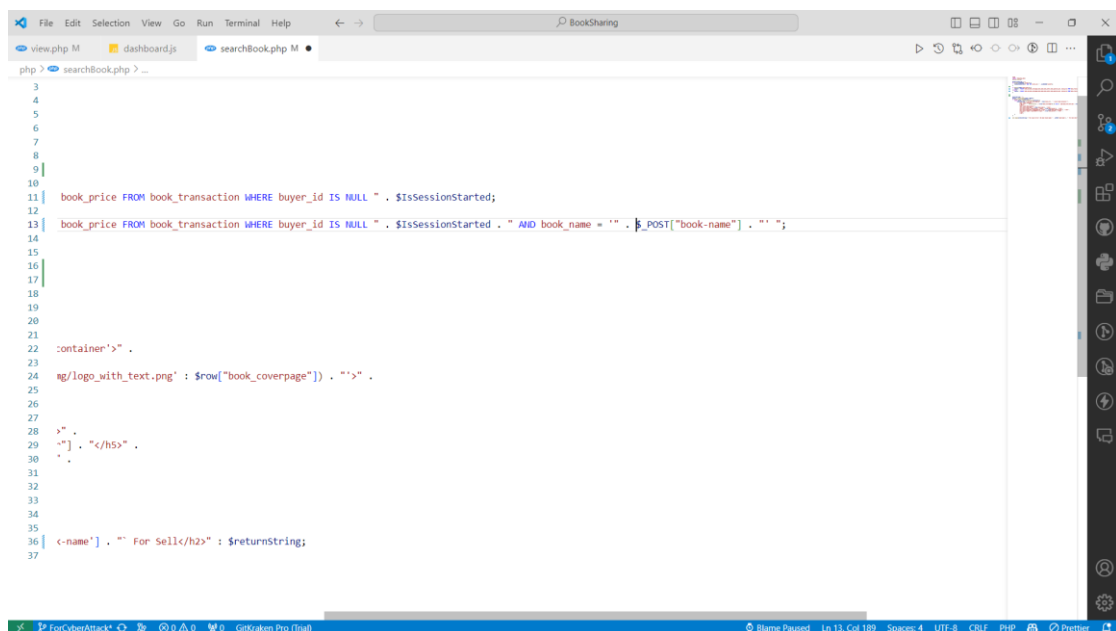
The screenshot shows the Burp Suite interface on the left and a web browser on the right. In Burp Suite, the 'Intercept' tab is active, showing the same request as before. The 'Inspector' tab shows the 'book-name' parameter, and the 'Value' field is highlighted. The value is `java' OR ' 1=1 --`. The web browser on the right shows the 'Dashboard - Book sharing' page with a search bar containing 'java'. Below the search bar, a message reads: 'No book found named `jav` For Sell'.

10) Now just click on forward and you will get all the book available in that website.

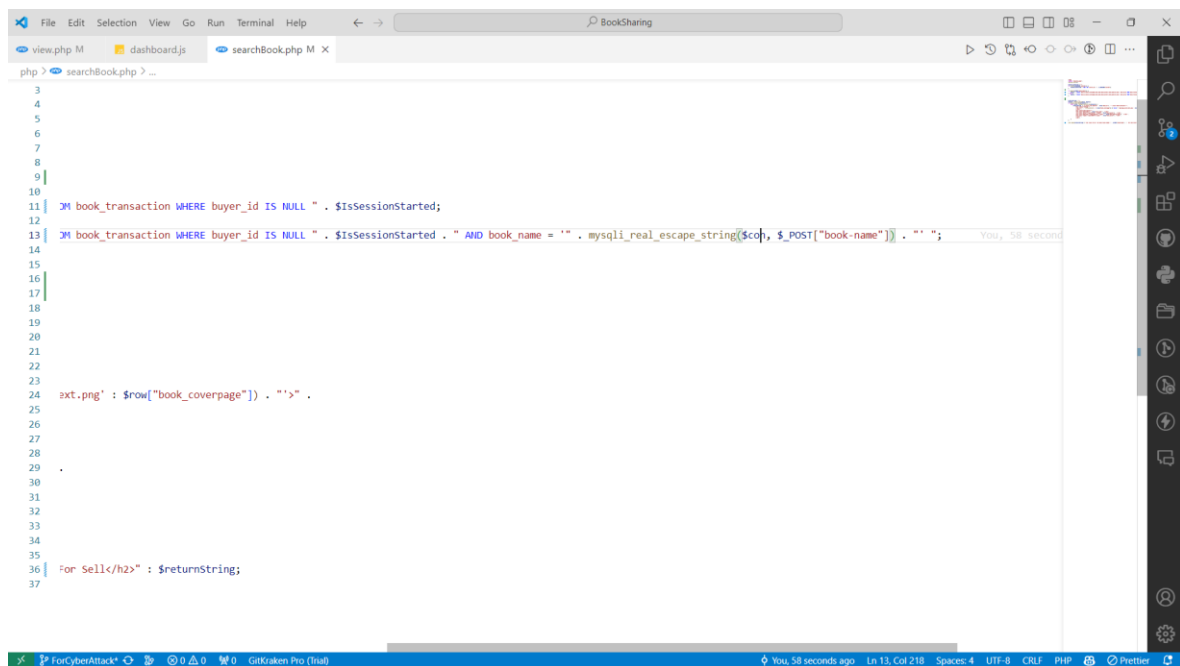


- **Solution for that Attack**

Code Before



Code After

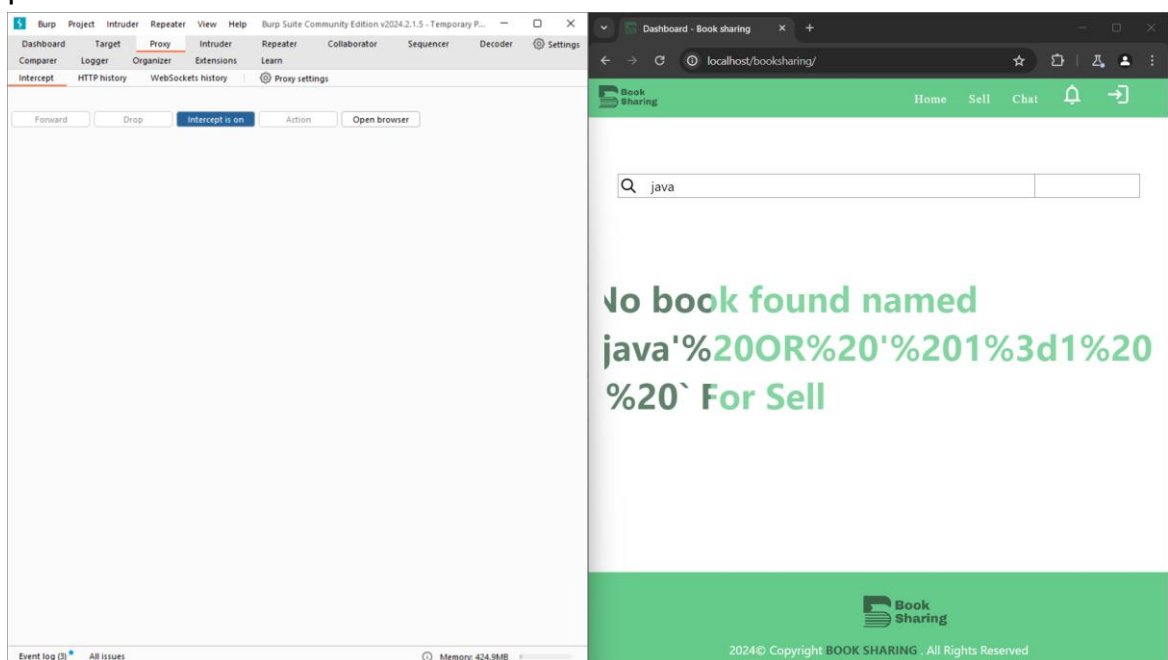


```

3
4
5
6
7
8
9
10
11 $ book_transaction WHERE buyer_id IS NULL " . $isSessionStarted;
12
13 $ book_transaction WHERE buyer_id IS NULL " . $isSessionStarted . " AND book_name = " . mysqli_real_escape_string($con, $_POST["book-name"]) . " ";
14
15
16
17
18
19
20
21
22
23
24 $xt.png" : $row["book_coverpage"]) . ">" .
25
26
27
28
29
30
31
32
33
34
35
36 for Sell/h2" : $returnString;
37

```

- 1) We have added the function named `mysqli_real_escape_string()`.
- 2) This function convert the all SpecialCharacter htmlEntities so there can't be attack possible.



- 3) We have done same thing but now the attack it not possible. Just because the expression `` java' OR ' 1=1 -- `` is now converted to whole string so now the attack can't be possible.

Experiment – 2

Aim: XSS (Cross-Site scripting).

- **What we had done for in this attack?**

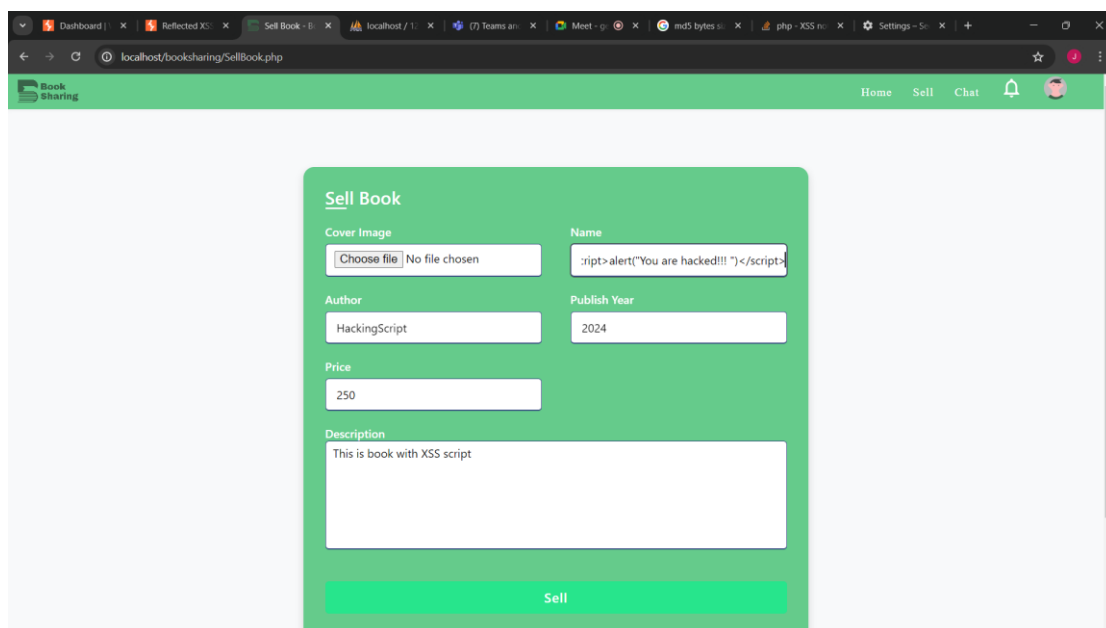
→ In this attack we had implemented the most basic attack of XSS. We have a planned to do BruteForce Attack on the site but we are unable to accommodate that thing into our website.

So, Finally we have came to conclusion to attack with most basic to demonstrate to the attack.

- **SQL Attack**

1) We have a website for selling and buying books.

2) There is a page named Sell book where we can sell the book by filliung out some necessary details.



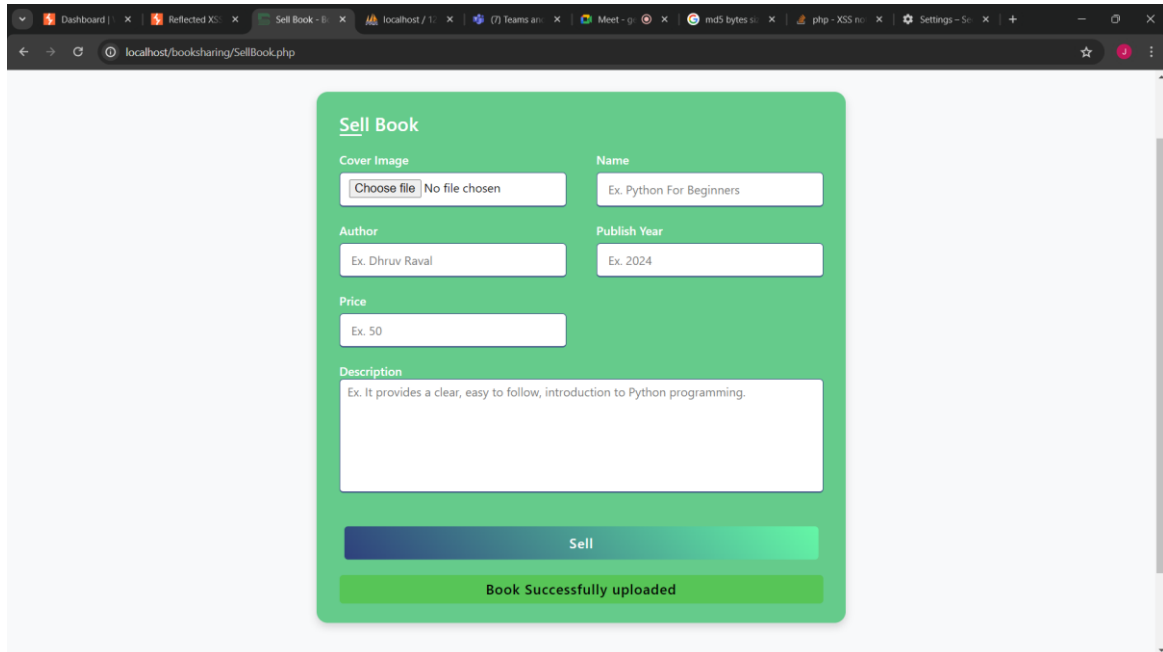
The screenshot shows a web browser window with multiple tabs. The active tab is 'localhost/booksharing/SellBook.php'. The website has a green header with 'Book Sharing' and navigation links: Home, Sell, Chat, and a user profile icon. The main content area features a 'Sell Book' form with the following fields:

- Cover Image:** A file selection button labeled 'Choose file' with the text 'No file chosen'.
- Name:** A text input field containing the JavaScript payload: `<script>alert('You are hacked!!!')</script>`.
- Author:** A text input field containing 'HackingScript'.
- Publish Year:** A text input field containing '2024'.
- Price:** A text input field containing '250'.
- Description:** A text area containing the text 'This is book with XSS script'.

A green 'Sell' button is located at the bottom of the form.

3) So, to start or do attack first of all we chosen this page cause from this the data is being stored on the database and will be loaded on a different page.

- 4) This will affect the website / run that script later on from that page is loaded.
- 5) One more thing that we have been discovered that that site is store anything from that input field so we have choose that specific input field.

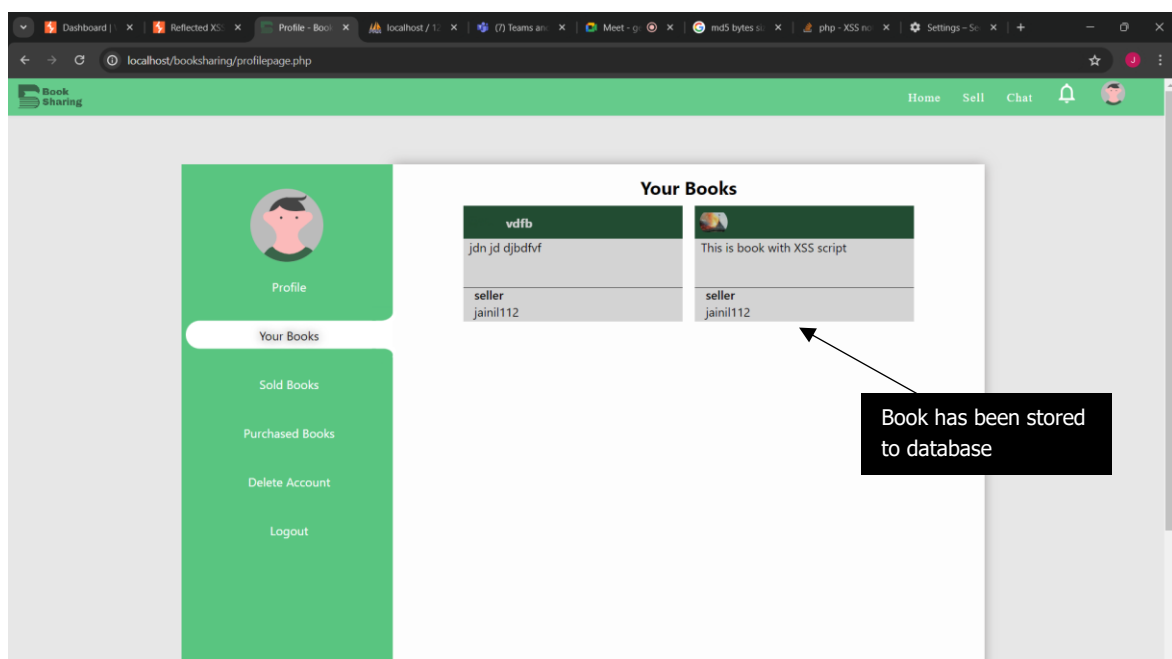


The screenshot shows a web browser window with the URL `localhost/booksharing/SellBook.php`. The page displays a 'Sell Book' form with the following fields:

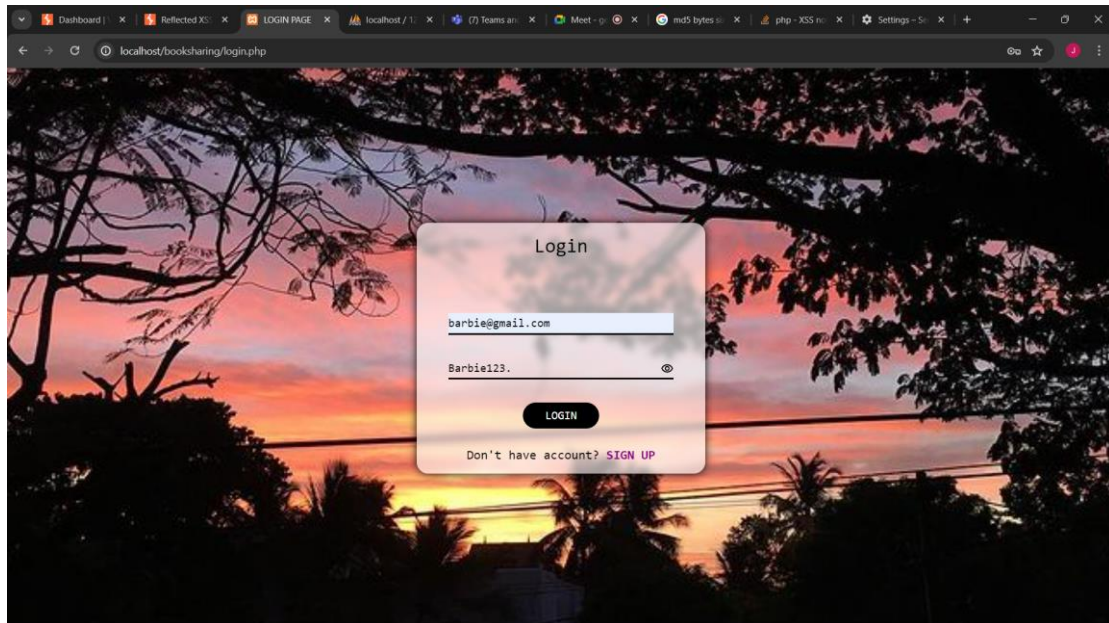
- Cover Image:** A file selection button labeled 'Choose file' with the text 'No file chosen'.
- Name:** A text input field with the placeholder text 'Ex. Python For Beginners'.
- Author:** A text input field with the placeholder text 'Ex. Dhruv Raval'.
- Publish Year:** A text input field with the placeholder text 'Ex. 2024'.
- Price:** A text input field with the placeholder text 'Ex. 50'.
- Description:** A large text area with the placeholder text 'Ex. It provides a clear, easy to follow, introduction to Python programming.'

At the bottom of the form is a green 'Sell' button. Below the button, a green message bar displays the text 'Book Successfully uploaded'.

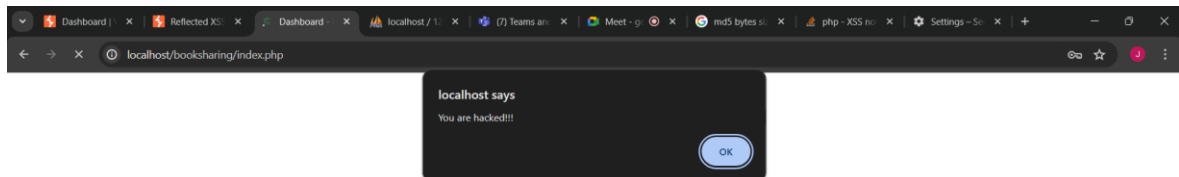
- 6) Letter on after submitting the form the book will be stored to the database and after that all other user will be shown a popup of that script have been injected to that field.



7) So, now we will login from another account to demonstrate the effect of that script.

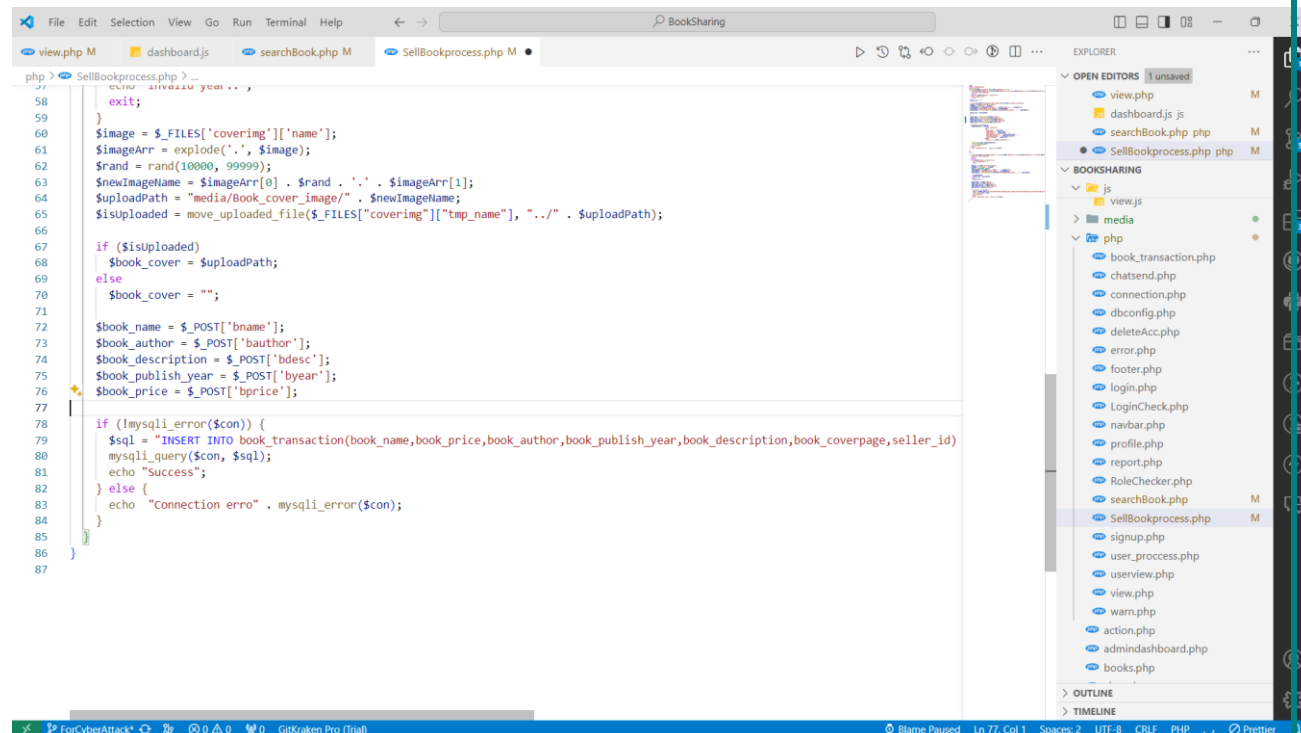


8) On the login user is landed on the index page where all books are shown. So, the moment that page is loaded the script file will be run and we will see the alert box



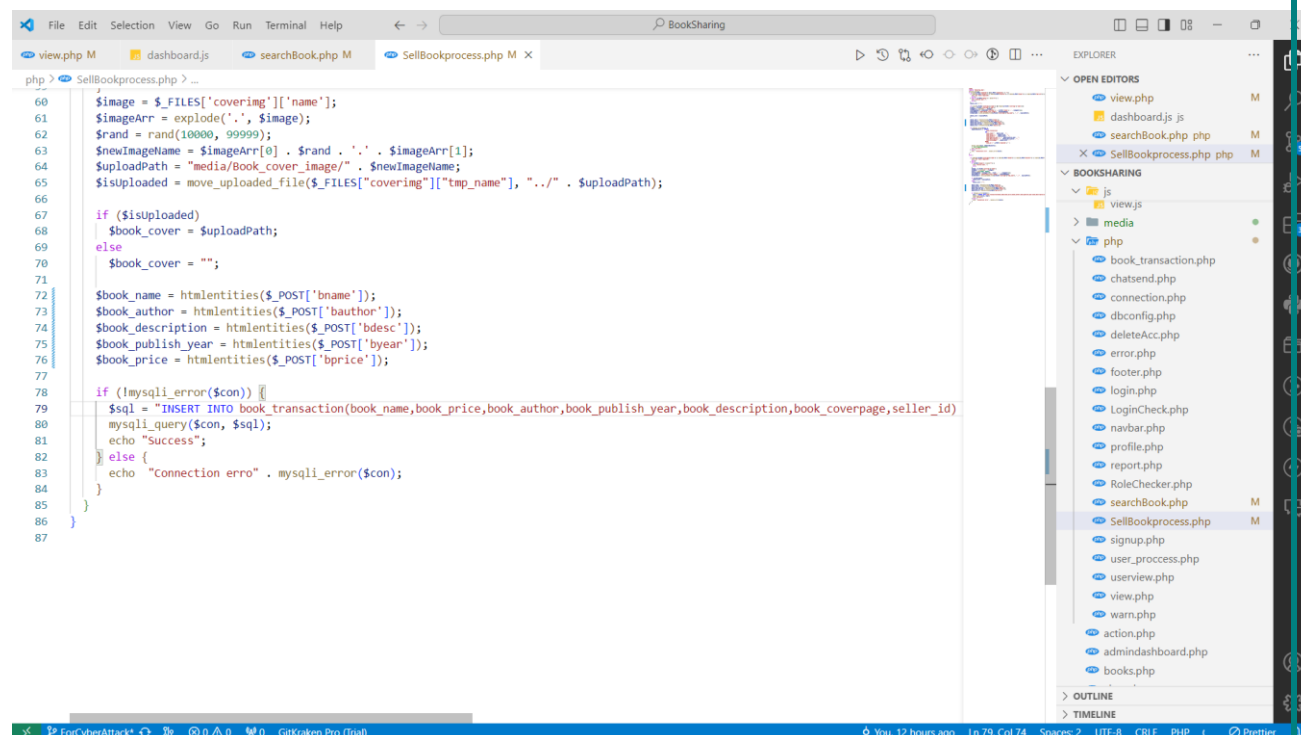
• Solution for that Attack

Code Before



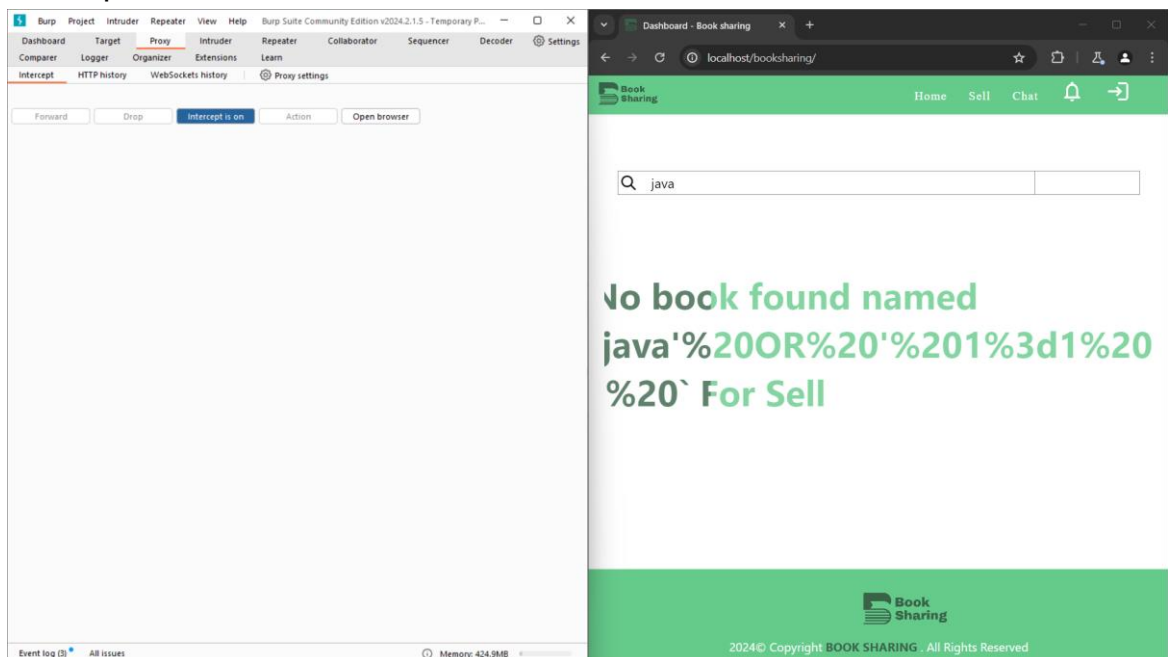
```
php > SellBookprocess.php > ...
58     exit;
59 }
60 $image = $_FILES['covering']['name'];
61 $imageArr = explode('.', $image);
62 $rand = rand(10000, 99999);
63 $newImageName = $imageArr[0] . $rand . '.' . $imageArr[1];
64 $uploadPath = "media/Book_cover_image/" . $newImageName;
65 $isuploaded = move_uploaded_file($_FILES["covering"]["tmp_name"], "../" . $uploadPath);
66
67 if ($isuploaded)
68     $book_cover = $uploadPath;
69 else
70     $book_cover = "";
71
72 $book_name = $_POST['bname'];
73 $book_author = $_POST['bauthor'];
74 $book_description = $_POST['bdesc'];
75 $book_publish_year = $_POST['byear'];
76 $book_price = $_POST['bprice'];
77
78 if (!mysqli_error($con)) {
79     $sql = "INSERT INTO book_transaction(book_name,book_price,book_author,book_publish_year,book_description,book_coverpage,seller_id)
80     mysqli_query($con, $sql);
81     echo "Success";
82 } else {
83     echo "Connection error" . mysqli_error($con);
84 }
85
86 }
87
```

Code After



```
php > SellBookprocess.php > ...
60 $image = $_FILES['covering']['name'];
61 $imageArr = explode('.', $image);
62 $rand = rand(10000, 99999);
63 $newImageName = $imageArr[0] . $rand . '.' . $imageArr[1];
64 $uploadPath = "media/Book_cover_image/" . $newImageName;
65 $isuploaded = move_uploaded_file($_FILES["covering"]["tmp_name"], "../" . $uploadPath);
66
67 if ($isuploaded)
68     $book_cover = $uploadPath;
69 else
70     $book_cover = "";
71
72 $book_name = htmlentities($_POST['bname']);
73 $book_author = htmlentities($_POST['bauthor']);
74 $book_description = htmlentities($_POST['bdesc']);
75 $book_publish_year = htmlentities($_POST['byear']);
76 $book_price = htmlentities($_POST['bprice']);
77
78 if (!mysqli_error($con)) {
79     $sql = "INSERT INTO book_transaction(book_name,book_price,book_author,book_publish_year,book_description,book_coverpage,seller_id)
80     mysqli_query($con, $sql);
81     echo "Success";
82 } else {
83     echo "Connection error" . mysqli_error($con);
84 }
85
86 }
87
```

- 4) We have added the function named htmlentities()
- 5) This function convert the all SpecialCharacter to htmlEntities so there can't be attack possible.



- 6) We have performed the same attack and added a book in a website with the script that but now that script is stop from being executed. Because the html tags and special character has been converted to HTML ENTITIES so now instade of render the whole script it will execute as a string so we can see the title of the book with script tag written on it.

