**TITLE** : Website Redirection using cross side scripting.

**ABSTRACT:**

The latest advancements in the internet and communication technology has made day-to-day life fully dependent on web applications as all the facilities are available at just one click. Network failure, data breaches, computer viruses and other malicious contents based Cyber-attacks have affected the security of Web applications and there is always a risk of information stealing or tampering. Website Redirection is one of the application layer vulnerabilities that targets web applications by embedding scripts in a web page that will get executed at client side or server-side and the attacker will manipulate the information in desired manner.

**INTRODUCTION :**

**Cross-Site Scripting (XSS):**

Cross-Site Scripting (XSS) is probably the most common singular security vulnerability existing in web applications at large. It has been estimated that approximately 65% of websites are vulnerable to an XSS attack in some form, a statistic which should scare you as much as it does me.

XSS occurs when an attacker is capable of injecting a script, often Javascript, into the output of a web application in such a way that it is executed in the client browser. This ordinarily happens by locating a means of breaking out of a data context in HTML into a scripting context - usually by injecting new HTML, Javascript strings or CSS markup. HTML has no shortage of locations where executable Javascript can be injected and browsers have even managed to add more. The injection is sent to the web application via any means of input such as HTTP parameters.

### 

### 

### **The consequences of malicious JavaScript**

Among many other things, the ability to execute arbitrary JavaScript in another user's browser allows an attacker to perform the following types of attacks:

**Cookie theft**

The attacker can access the victim's cookies associated with the website using document.cookie, send them to his own server, and use them to extract sensitive information like session IDs.

**Keylogging**

The attacker can register a keyboard event listener usingaddEventListener and then send all of the user's keystrokes to his own server, potentially recording sensitive information such as passwords and credit card numbers.

**Phishing**

The attacker can insert a fake login form into the page using DOM manipulation, set the form's action attribute to target his own server, and then trick the user into submitting sensitive information.

**Methods of preventing XSS**

In XSS, user input is mistakenly interpreted as malicious program code. In order to prevent this type of code injection, secure input handling is needed. For a web developer, there are two fundamentally different ways of performing secure input handling:

* **Encoding**, which escapes the user input so that the browser interprets it only as data, not as code.
* **Validation**, which filters the user input so that the browser interprets it as code without malicious commands.

**IMPLEMENTATION DETAILS:**

**product.php:**

This site is similar to any e-commerce shopping portal viz. Amazon, Flipkart etc. This page describes all the details with respect to the product and also shows the product reviews. When logged in it allows the user to enter his own reviews with respect to the product.

It contains 2 parts:

Php: It fetches the reviews of the users with respect to the product and passes it to html for rendering.

Html: It gives us the product description and displays the reviews of other users.It also gives us an option to comment.

<?php

$conn=mysql\_connect('localhost','root',"");

mysql\_select\_db("ntal",$conn);

$query = "SELECT \* from ntal";

$res = mysql\_query($query,$conn);

?>

<html>

<body style="background-color:#FFFFFF;">

<center> <h2>Nikon D5200 with (AF-S 18 - 55 mm VR II + AF-S 55 - 200 mm ED VR II Kit Lens) DSLR Camera (Black)</h2></center>

<center>

<img src="https://rukminim1.flixcart.com/image/832/832/camera/r/j/j/nikon-d5200-with-af-s-18-55-mm-vr-kit-nikon-af-s-dx-vr-zoom-original-imadte3wghzsfczs.jpeg?q=70" width="400px"/><br>

</center>

<center> <h2><b>Rs. 21,999/-</b></h2>

<br>

<input type="button" value="BUY" style="background-color:Yellow;font-size:22px" ></input>

</center>

<br>

<h2>Comments</h2>

<br>

<?php

while($row = mysql\_fetch\_array($res, MYSQL\_ASSOC)) {

echo "<font style='font-size:20px'>\"". $row['comment'] . "\"</font><br><br>";

echo " <font style='font-size:20px'>-".$row['name']."</font><br><br>";

}

?>

<h2>Please enter your valuable comment</h2><br>

<form action= "put\_comment.php" method="post">

<table>

<tr>

<td>Name:</td>

<td><input type="text" id="name" name="name" required /></td>

</tr>

<tr>

<td>Comment:</td>

<td><textarea rows="10" cols="40" id="comment" name="comment" required></textarea></td>

</tr>

</table>

<input type="submit" value="submit" style="font-size:22px">

</form>

</body>

</html>

**add\_comment.php:**

product.php page passes the comment entered by the user to add\_comment.php.It then connects to the database and then stores the username and the comment in the database.

<?php

$n=$\_POST['name'];

$c=$\_POST['comment'];

$servername="localhost";

$usern="root";

$conn=mysql\_connect($servername,$usern,"");

if(!$conn){

die("Connection failed: ".mysql\_error());

}

mysql\_select\_db("ntal",$conn);

$sql="insert into ntal(`name`,`comment` ) values ('$n','$c')";

echo $sql;

header('Location: asd.php');

?>

If in the comment section a script tag:

<script> window.location="<https://www.jungleerummy.com/>"; </script> for redirection is entered by a hacker, it is stored as it is in the database. While fetching the comments from the database the browser renders the script tag instead of displaying it.This leads to redirection from the product page to the hackers site.

**Prevention**

Redirection to another site occurs because the comment entered by the user is not examined for any malicious code and is directly stored in the database.To prevent website redirection we filter out the malicious content and display only the text.Thia can be achieved by using the **strip\_tags**  function in php.Strip\_tags function filters out all the html tags in the string given to it.

**RESULT:**

We observed that Cross-Site Scripting (XSS) is probably the most common singular security vulnerability existing in web applications. We have thus demonstrated one of the possible attacks in XSS and provided a possible solution.

**SNAPSHOTS:**

The following are the snapshots of the project

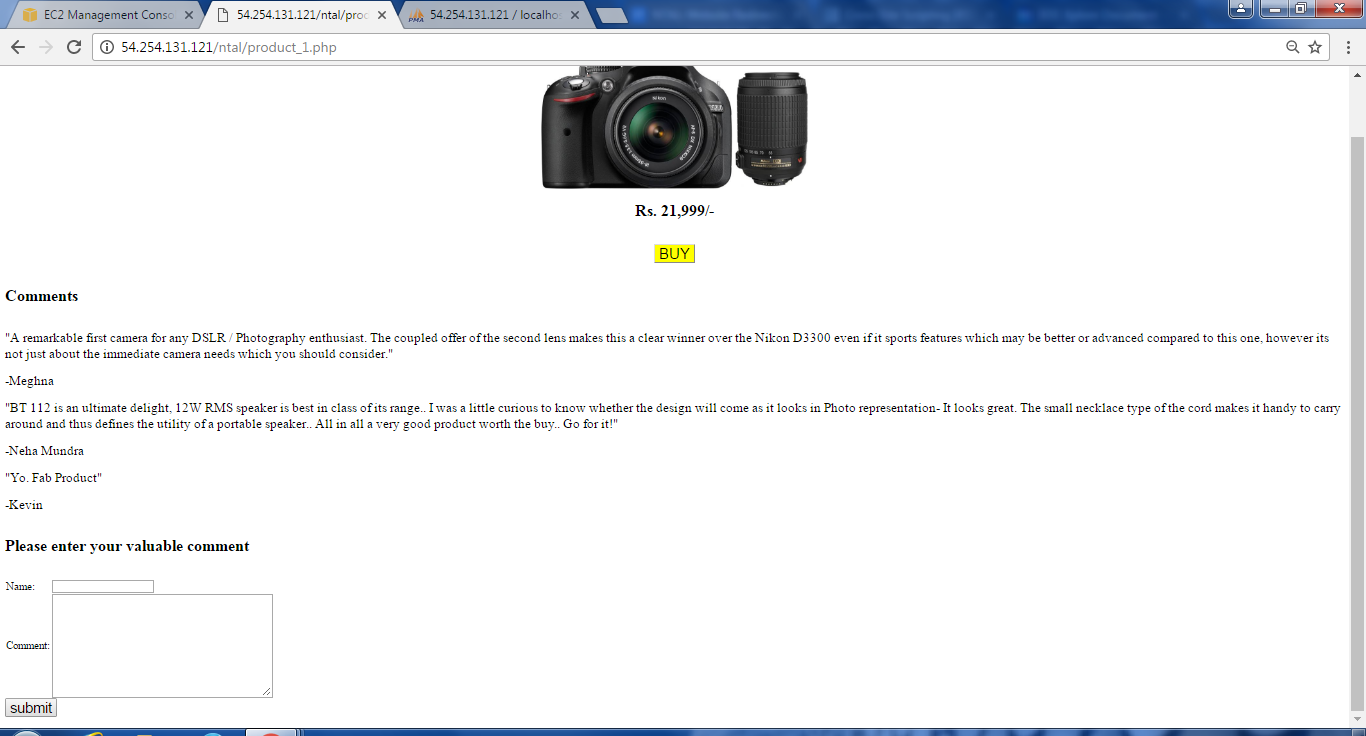


Fig 1:Product Description page

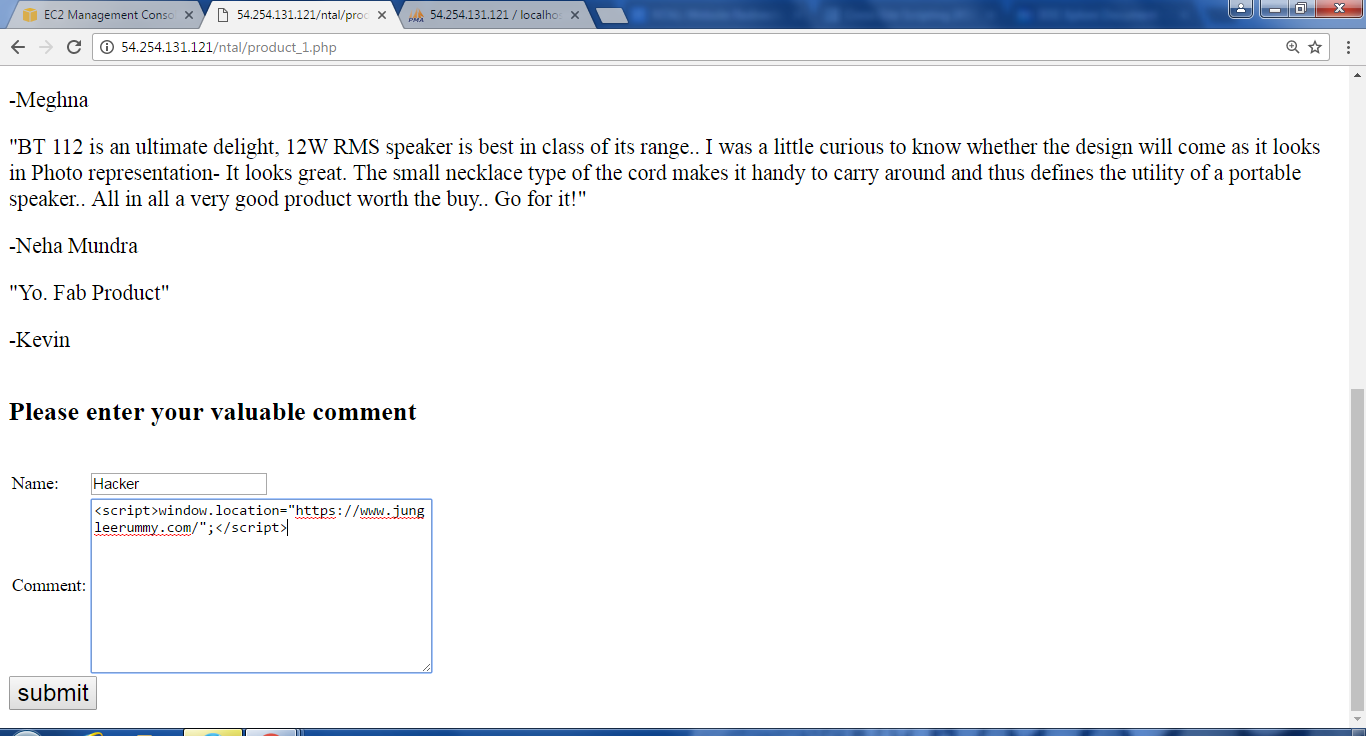


Fig 2:Hacker adding a script tag

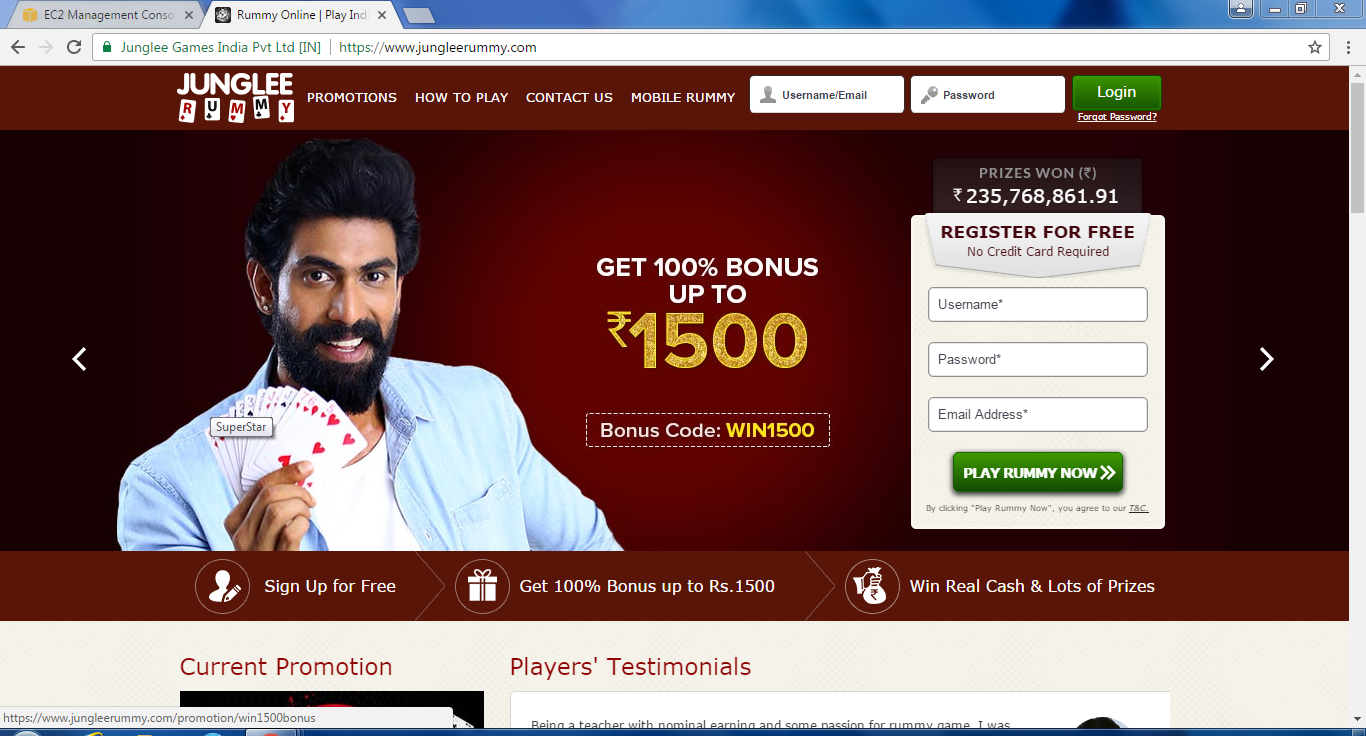


Fig 3: Genuine user redirected to hacker’s page

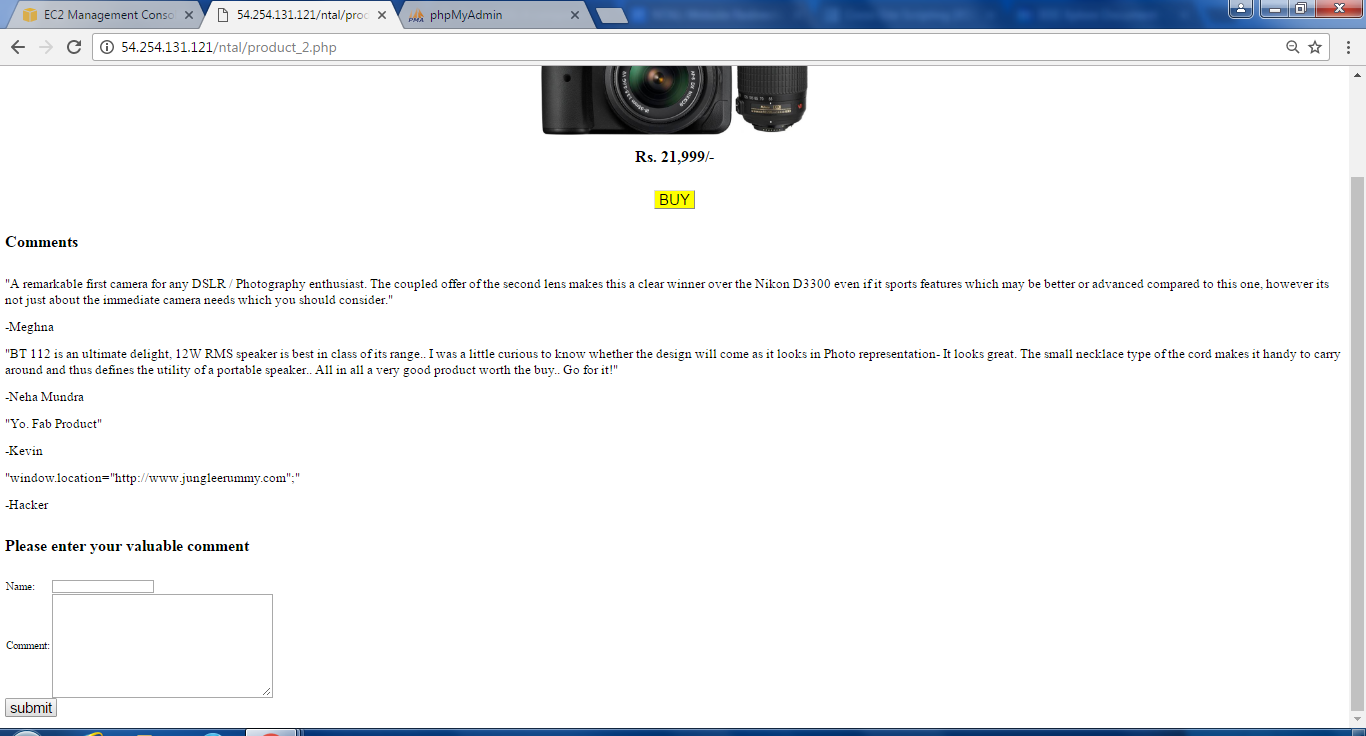


Fig 4: Prevention by using strip\_tags()



Fig 5:Script tag filtered by strip\_tag()

**REFERENCES**

<http://phpsecurity.readthedocs.io/en/latest/Cross-Site-Scripting-(XSS).html#external-application-defenses>

<http://excess-xss.com/>

An Automaton based Approach for forestalling Cross Site Scripting attacks in web application- <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7562786>

Impact Analysis of Preventing Cross Site Scripting and SQL Injection Attacks on Web Application- <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7456668>