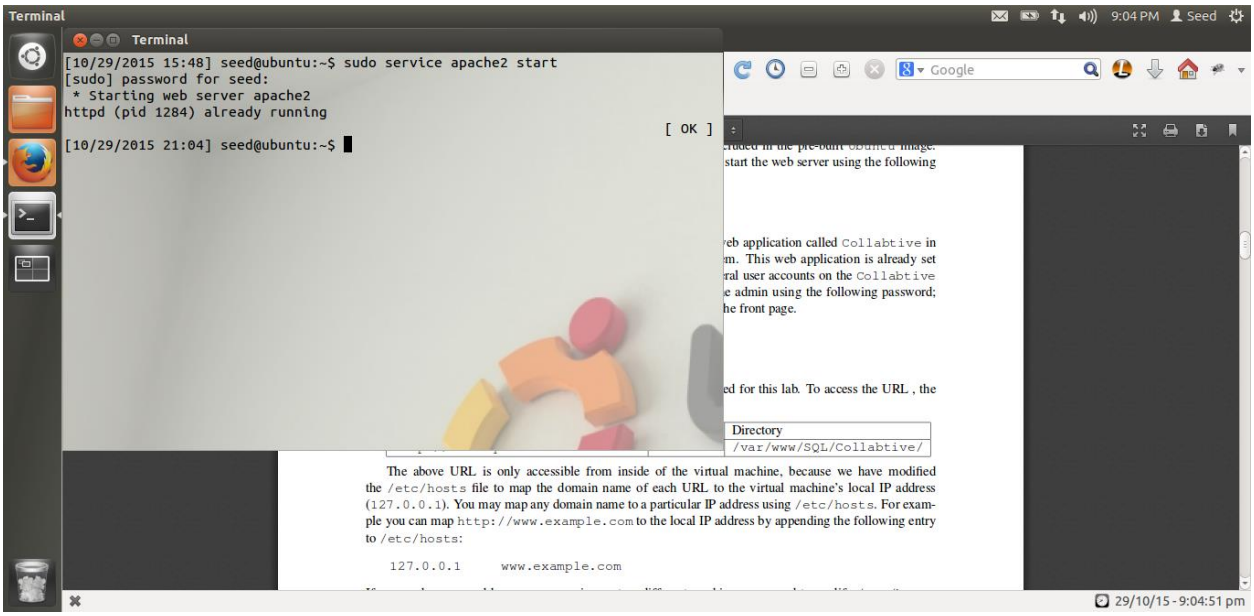
Yogesh Chaudhari
SUID: 244195971

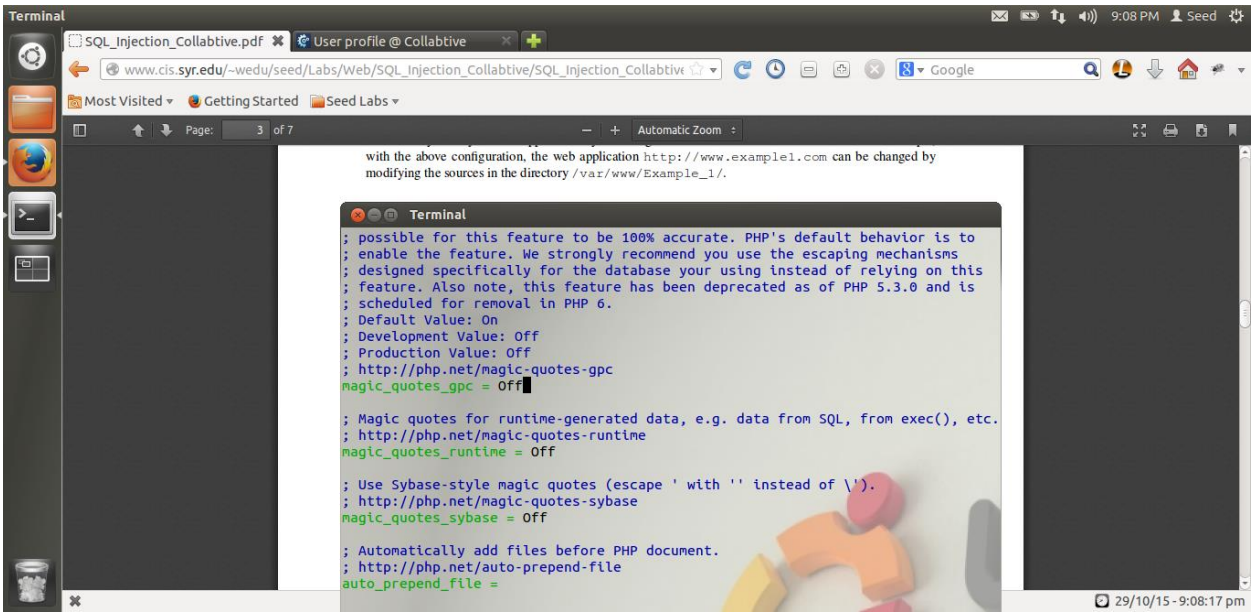
Initial Setup:

Starting apache server:



Here we are starting apache server using **service apache2 start** command.

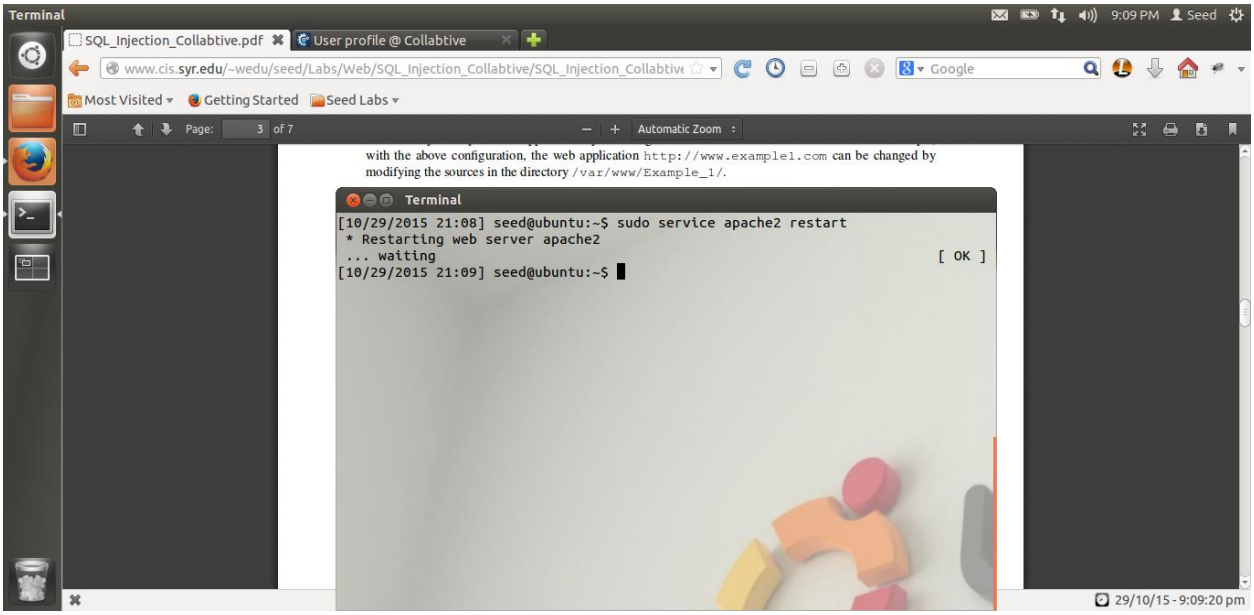
Turning off protection:



Now, here we are turning off the protection mechanism of our Collabtive application using **magic_quotes_gpc = off** command.

Yogesh Chaudhari
SUID: 244195971

Restart apache server:

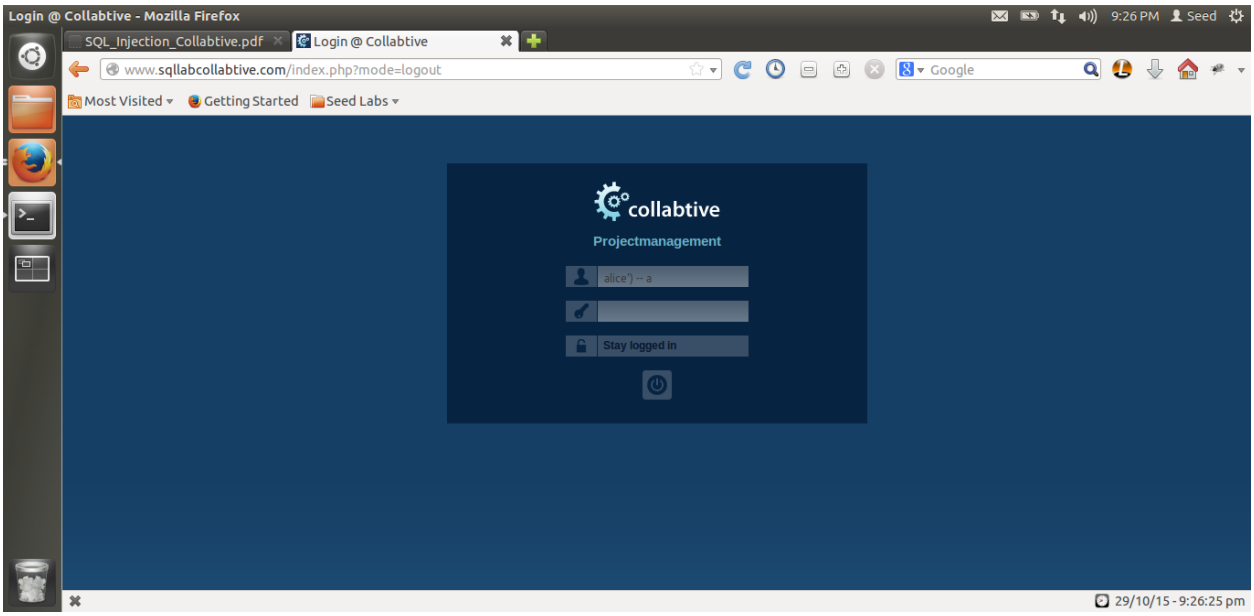


Here we are restarting apache server to make sure that our protection mechanism is turned off.

Yogesh Chaudhari
SUID: 244195971

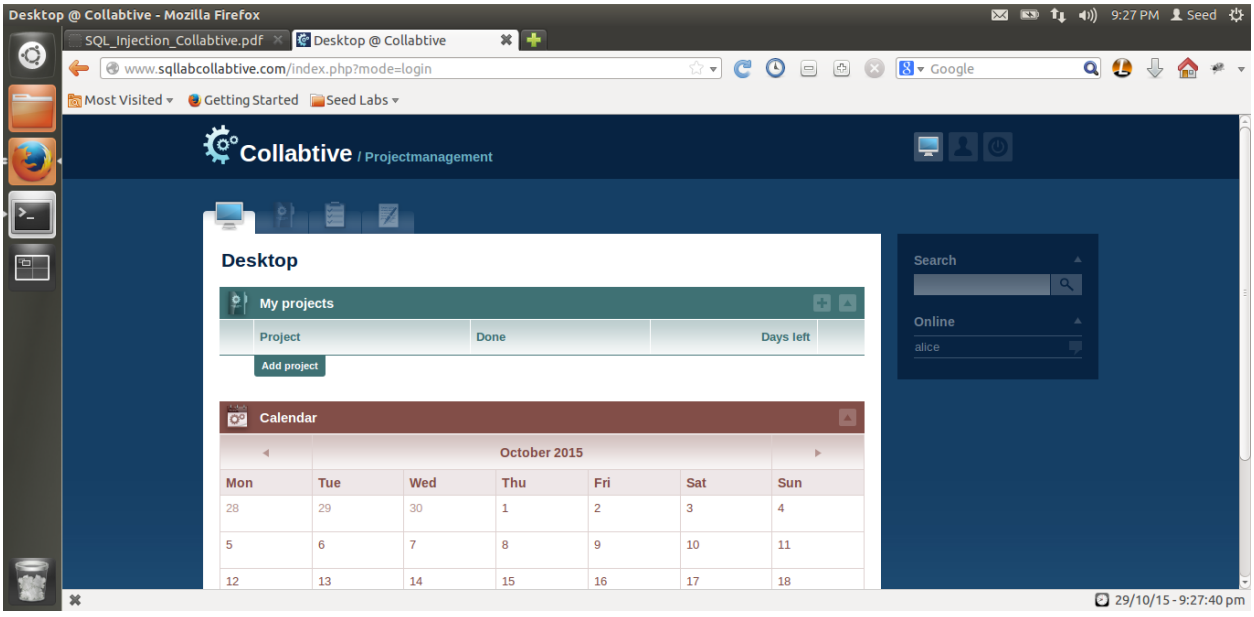
Task 1.1:

Attack on Alice's account:



Here we are using **alice') – a** commad to attack on alice's profile.

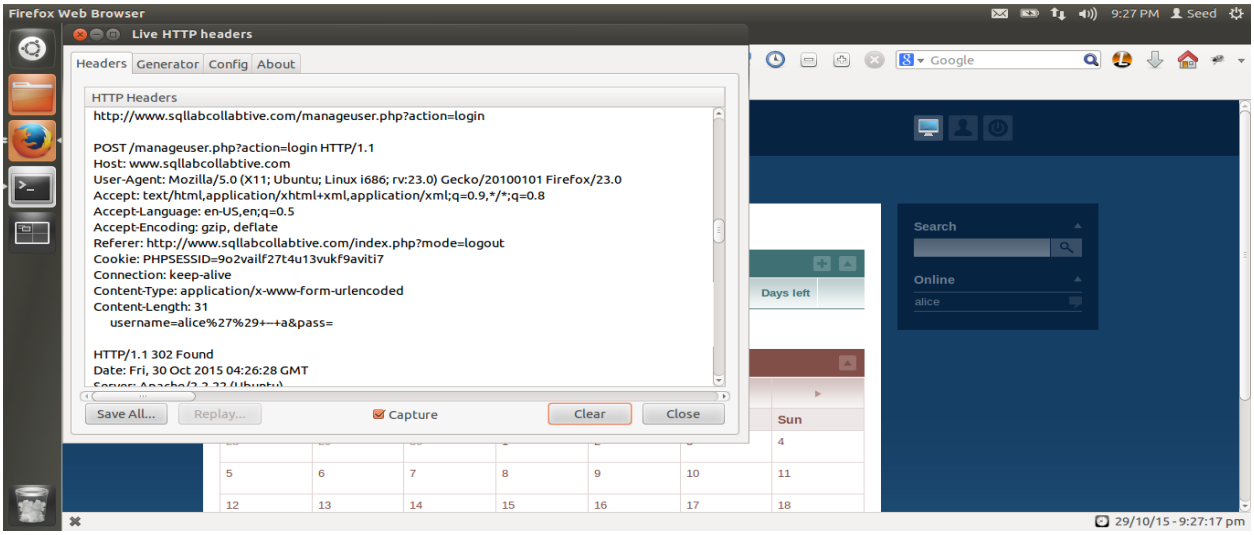
Attack Successful:



From above screen-shot we can see that our attack was successful and we were able to log in to the Alice's profile without knowing her password.

Yogesh Chaudhari
SUID: 244195971

Attack Successful live HTTP Header:



Here we can see that the string we sent back to the server in user input field is **alice') -- a**

and password input field is empty.

The SQL statement which is used on server side for validation is

SELECT ID, name, locale, lastlogin, gender FROM USERS_TABLE WHERE (name = '$user' OR email = '$user') AND pass = '$pass'

So in above statement when we put our string value in place of $user variable we get following SQL statement:

SELECT ID, name, locale, lastlogin, gender FROM USERS_TABLE WHERE (name = 'alice') -- aOR email = '$user') AND pass = '$pass'

Hence, in above statement everything after – will be considered as comment in SQL. Here while sending this string value as input value to server we have to make sure that after – we get a space character ' ' and then next part of SQL staement according to SQL standards, so we append – characters with space chaeacter ' ' and garbase string or character value 'a'. If we do not put any garbage string after – and space character, the request won't send the space character back to server and we will get Syntax error in SQL query at the server side. And the first part before – will select the user's ID, name, locale, lastlogin and gender details from USER_TABLE when name of the user is 'alice'. As the server will find a data which satisfies the SQL statement given above, it sends back the result to php file.

if (found one record)

then {allow the user to login}

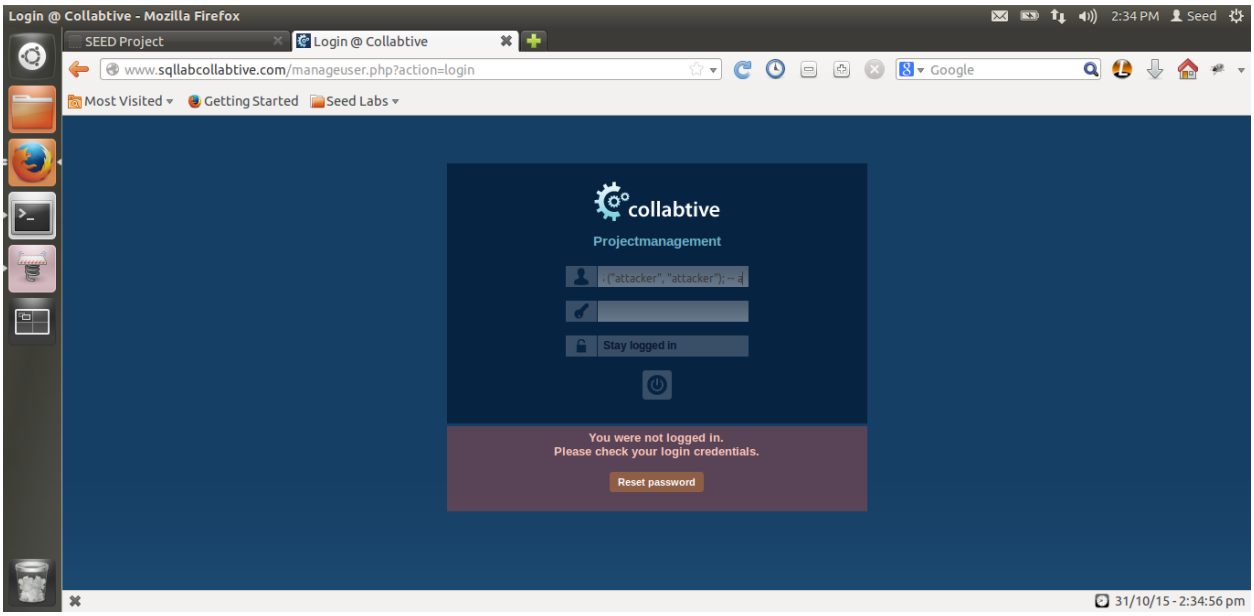
Hence, the above if command condition will be true as we found one record on server side and hence the user will be allowed to login into the account.

And hence our attack was successful.

Yogesh Chaudhari
SUID: 244195971

Task 1.2:

Insert Satement:



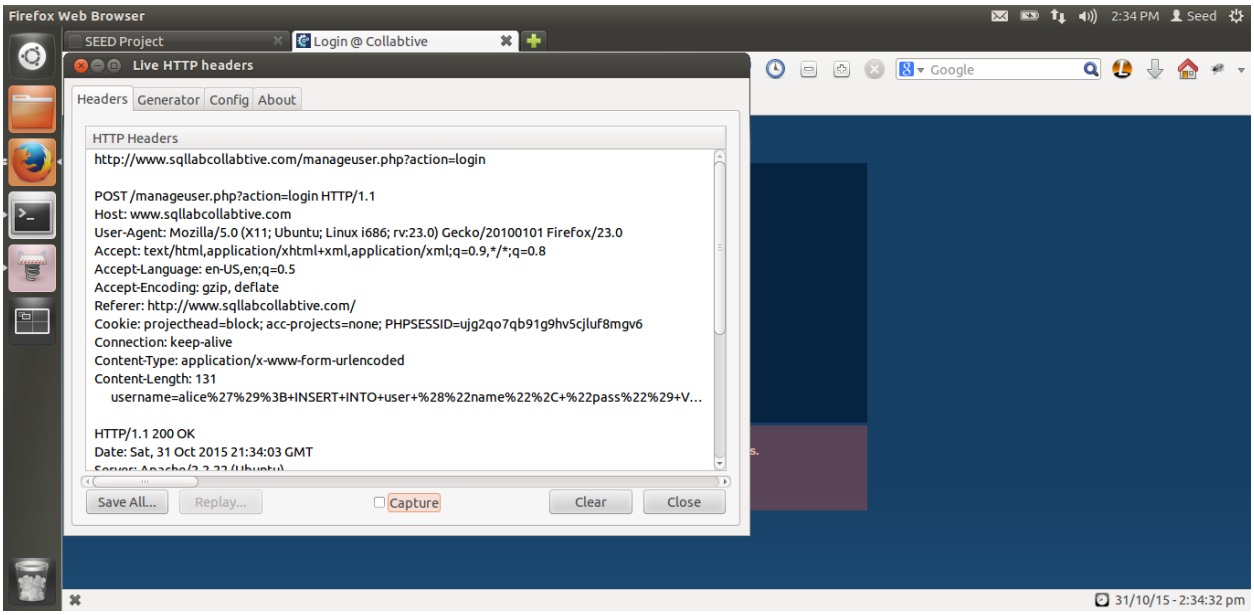
Here we using the same concept as explained in above task to write the statement in user input field which will insert a new user information into the database of collabtive application.

The statement used: In User filed:

alice'); INSERT INTO user (name,pass) VALUES ('attacker','attacker') -- a

Attack Unsuccessful:

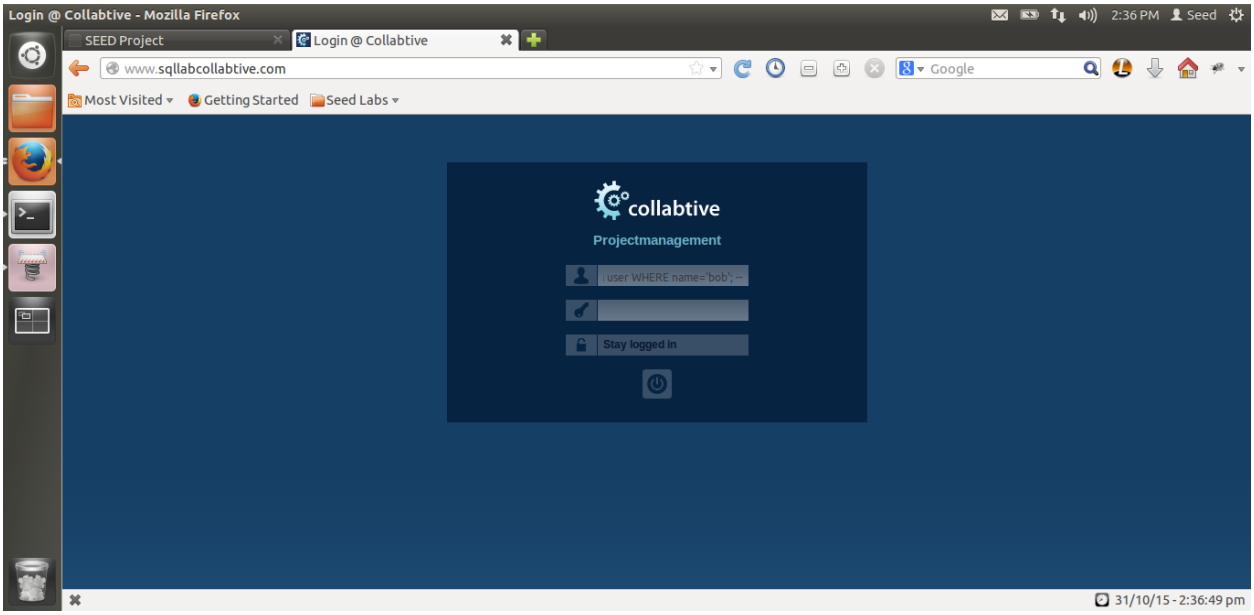
Live HTTP Header:



Yogesh Chaudhari
SUID: 244195971

From above screenshot we can see that our attack was unsuccessful. This is because of the use of mysql_query() function used to send the query on the server side. This function allows to send only unique SQL query inside it to the active server database and produces result for that. If two queries are inserted into this function it doesnot send them to the server. Hence, our attack was unsuccessful.

Delete Statement:



Here we using the same concept as explained in above task to write the statement in user input field which will delete user information from the database of collabtive application.

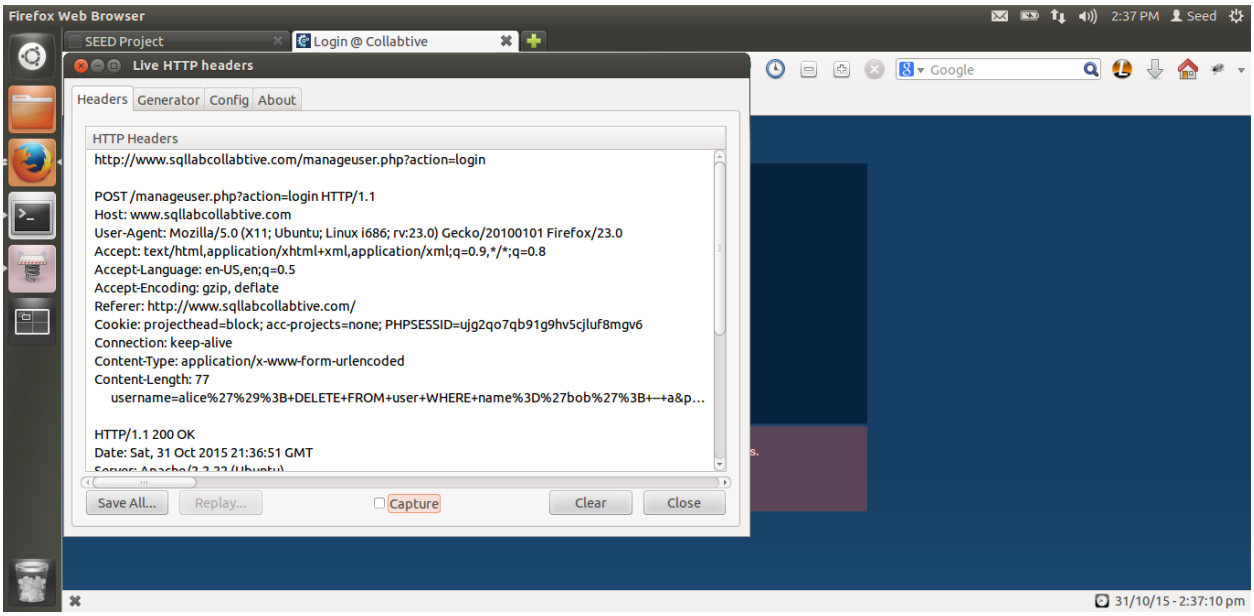
The statement used: In User filed:

alice'); DELETE FROM user WHERE name='bob' -- a

Yogesh Chaudhari
SUID: 244195971

Attack Unsuccessful:

Live HTTP Header:

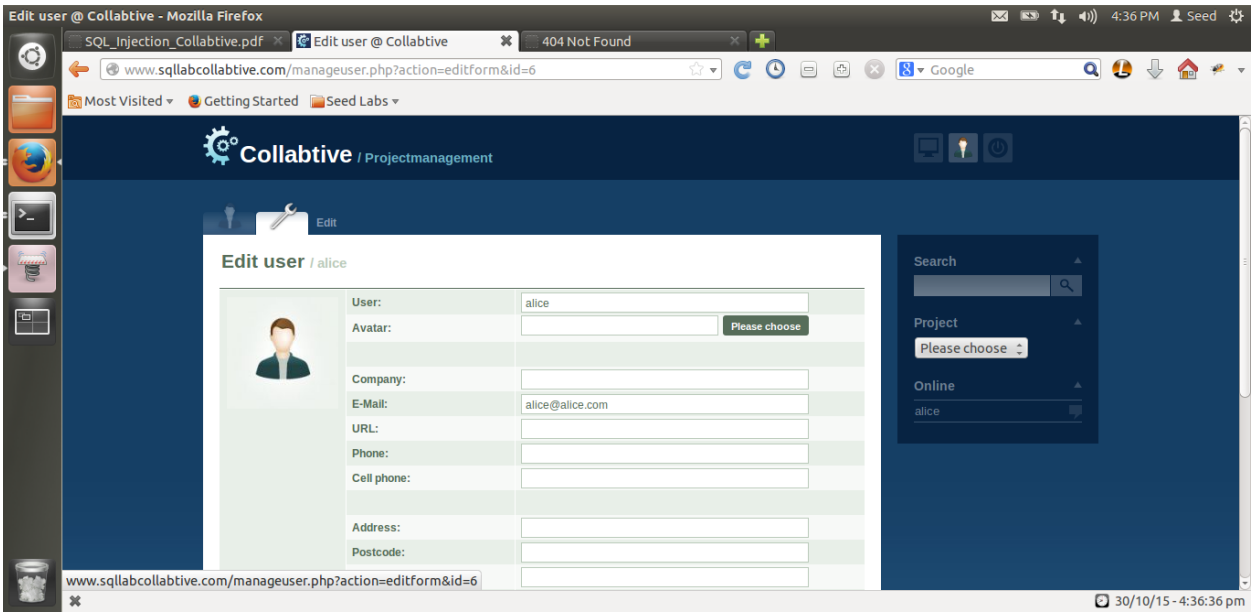


From above screenshot we can see that our attack was unsuccessful. This is because of the use of mysql_query() function used to send the query on the server side. This function allows to send only unique SQL query inside it to the active server database and produces result for that. If two queries are inserted into this function it doesnot send them to the server. Hence, our attack was unsuccessful.

Yogesh Chaudhari
SUID: 244195971

Task 2:

In Alice's Profile:



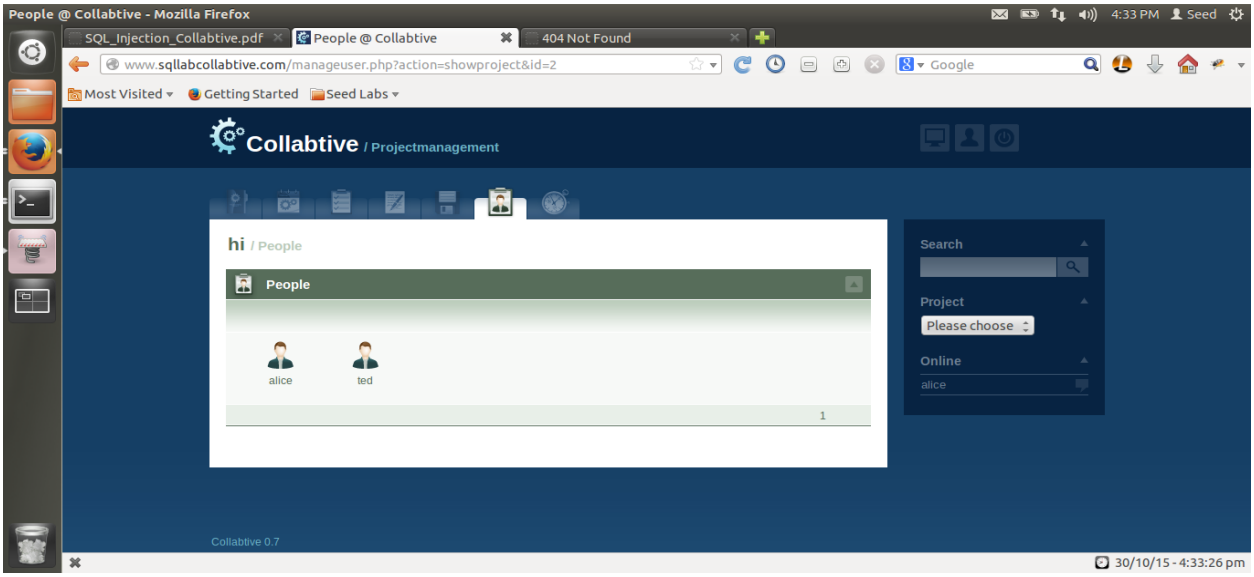
Here we have logged in to the Alice's profile. Here from above screenshot we can see that for displaying the online users the user's username is used, while in profile the username of user is used to display the profile of that specific user (e.g. User Profile/user_name) and also in profile URL we have id field which is the database unique ID value for each user.

Yogesh Chaudhari
SUID: 244195971

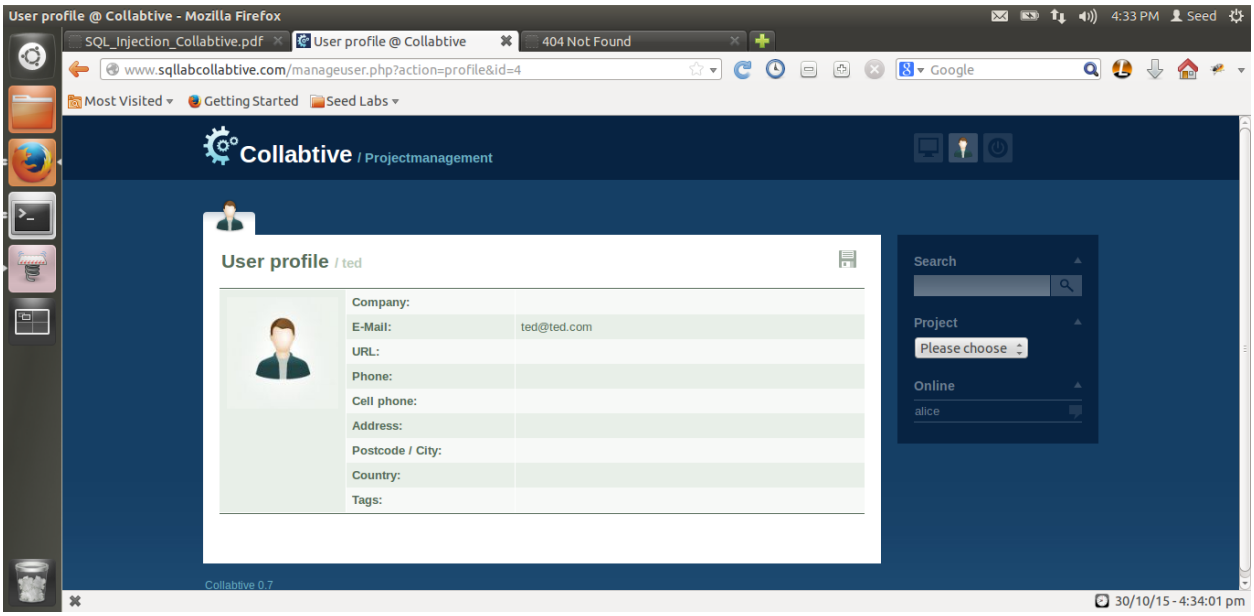
Getting Ted's username or ID value:

Method 1:

Here I have added a new project to Alice's profile named 'hi' and while creating a new project we get to choose the users with whom we want to share that project in that field we can select Ted from dropdown list. Now, Ted has been added as a user to 'hi' project.



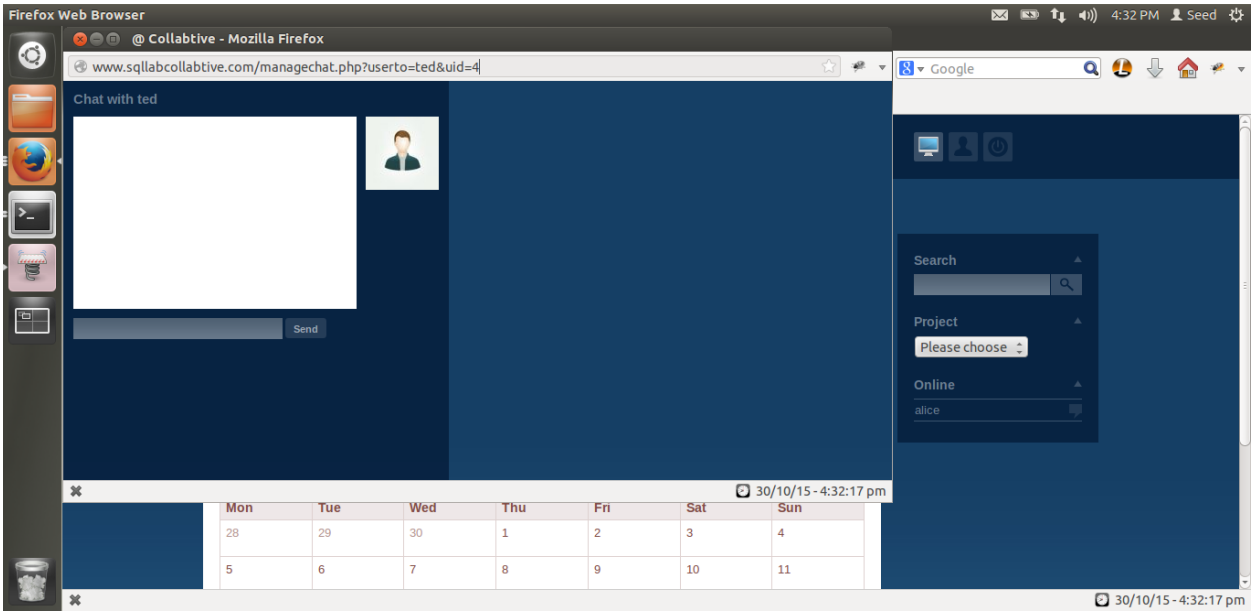
Here in above screenshot we can see that the users are displayed with their username and hence, we get Ted's username as 'ted'.



Also, when we select the ted user the application displays Ted's profile and here we can see that the username of Ted is 'ted' and from URL we can see that id of Ted is 6.

Yogesh Chaudhari
SUID: 244195971

Method 2:

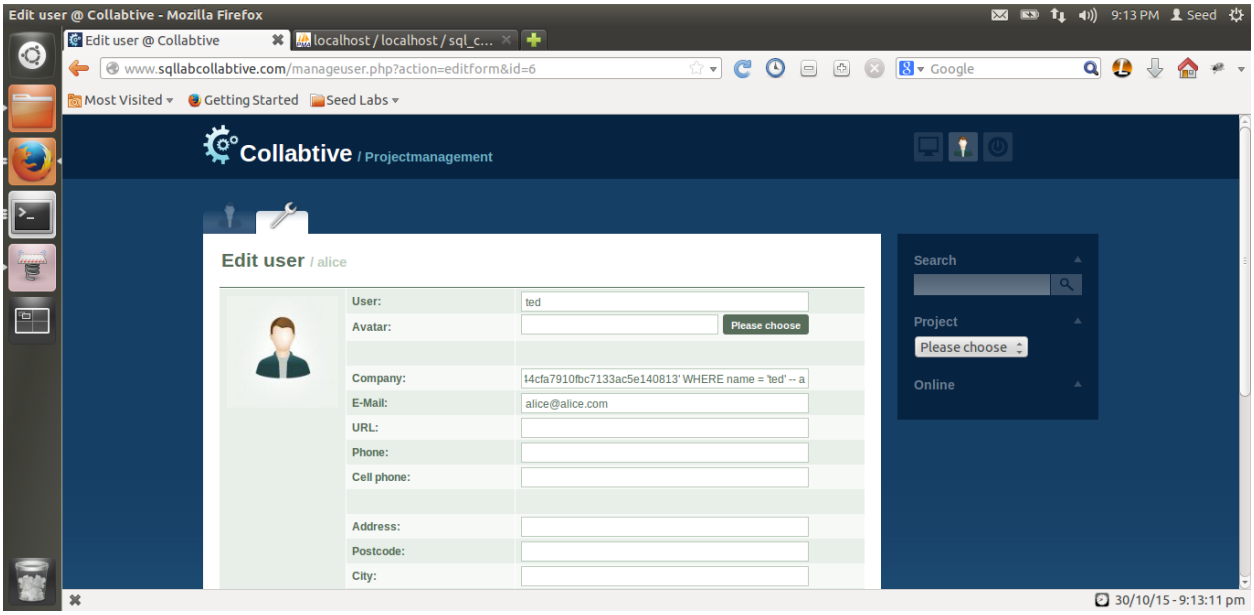


Here we are selecting message to Ted option from Alice's profile and a new window gets popped up.

In this new message window in URL section we can see that, the username of Ted is 'ted' and user_ID is 6 which are given as value to userto and uid variables respectively in URL.

Yogesh Chaudhari
SUID: 244195971

Attacking Ted's Profile using SQL Injection:



Now by using the username we have got from above two methods we can create a new SQL query which will replace Ted's password with password 'attack' using SQL injection attack from Alice's profile.

Attack statement used,

', pass='sha1_function_value_of_string_attack' WHERE name = 'ted' – a

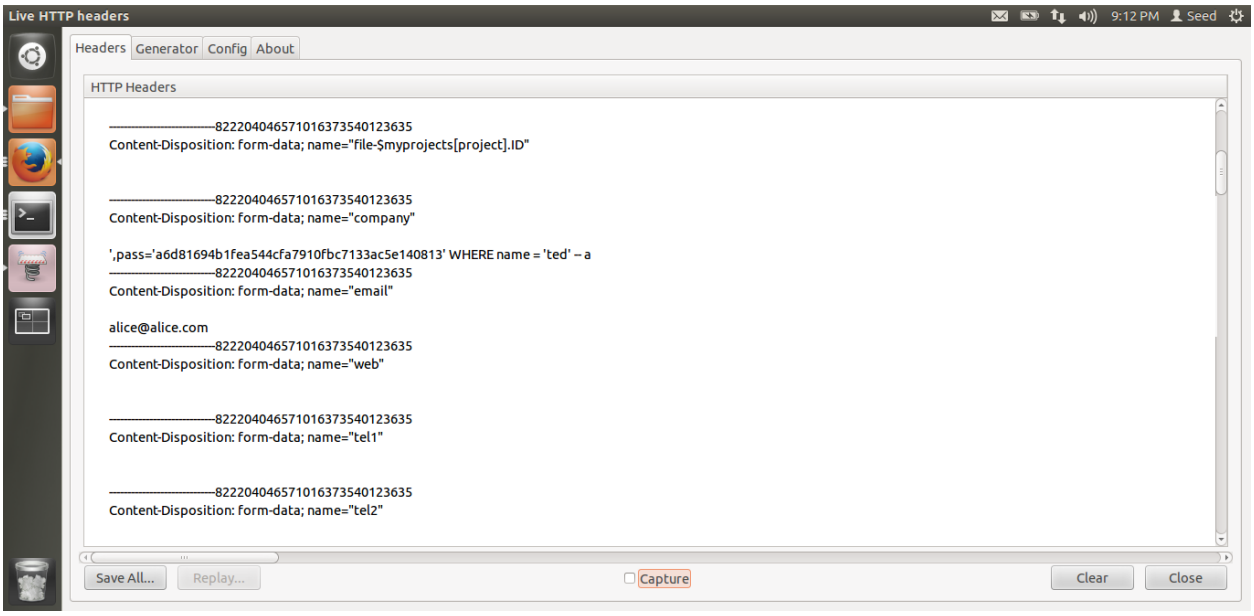
OR we can also use

', pass='sha1_function_value_of_string_attack' WHERE ID = '6' -- a

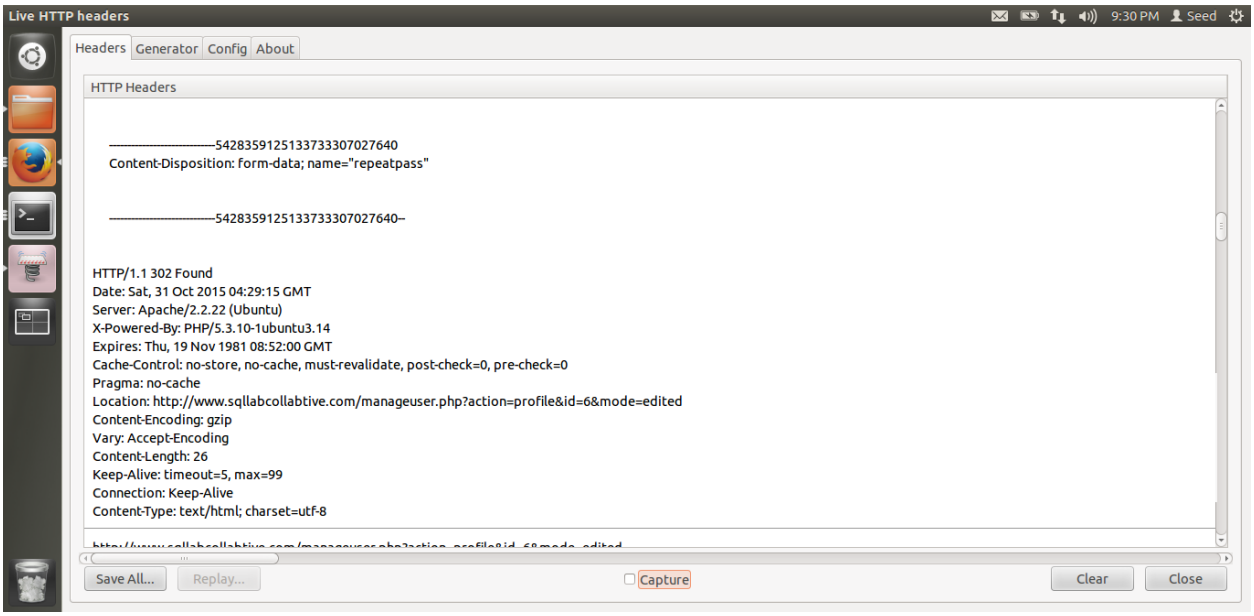
We have written this statement in 'company' input field of Alice's profile which is SQL Injection attack prone input field.

Here we need to use 'ted' as value in User input field or some other value because the way we are attacking on Ted's profile his username will also get updates. Hence, it should be 'ted' or some other unique string value not present in database user name column.

Yogesh Chaudhari
SUID: 244195971



Attack Request Successful:



From above screenshot we can see that our attack was successful. The SQL statement used on server side for edit profile is given as follows,

UPDATE user SET name='$name',email='$email', tel1='$tel1', tel2='$tel2', company='$company',zip='$zip',gender='$gender',url='$url',adress='$address1',adress2='$address2',state='$state',country='$country',tags='$tags',locale='$locale',rate='$rate' WHERE ID = $id

In this statement our attack statement will be placed in the place of $company variable. Hence, after substituting the value SQL statement at server side will be,

Yogesh Chaudhari
SUID: 244195971

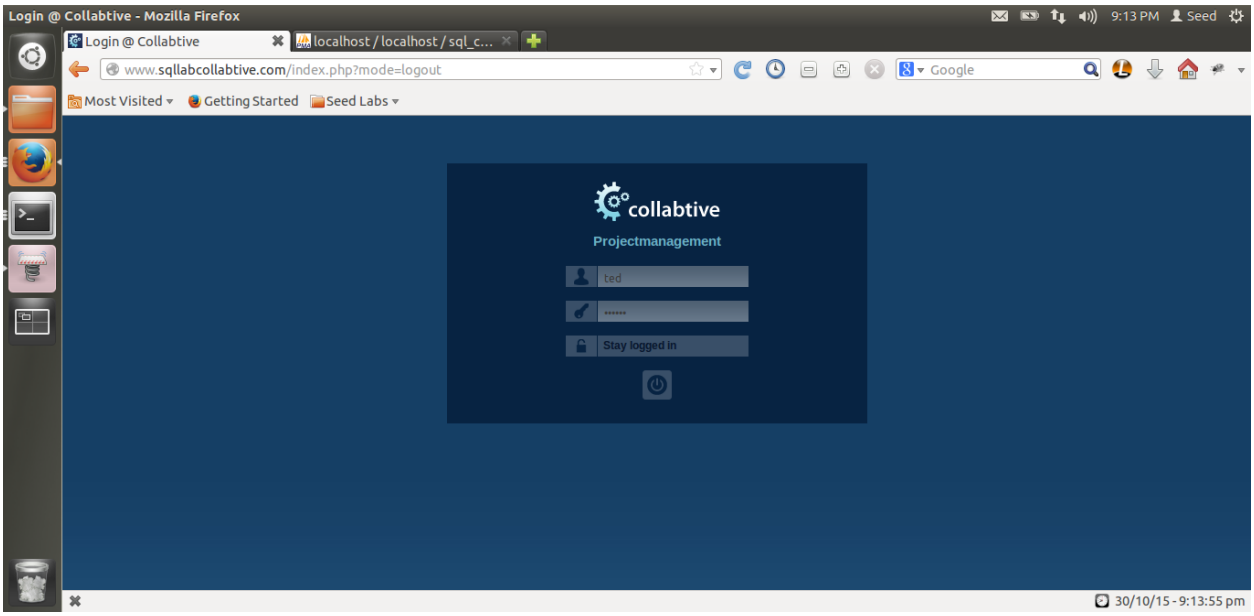
UPDATE user SET name='$name',email='$email', tel1='$tel1', tel2='$tel2', company='', pass=' a6d81694b1fea544cfa7910fbc7133ac5e140813' WHERE name = 'ted' -- a',zip='$zip',gender='$gender',url='$url',adress='$address1',adress2='$address2',state='$state',country='$country',tags='$tags',locale='$locale',rate='$rate' WHERE ID = $id

Here Ted's profile data from database will get selected due to SQL command "WHERE name='ted'" and its name, email, tel1, tel2, company, pass fields of Ted's data will be updated due to UPDATE SQL command. These fields will get the values as specified in the Alice's profile input fields. Everything after ' – ' will be considered as comment in SQL.

Here we can either use name='ted' as condition for WHERE clause or also we can use id='6' as condition for WHERE clause which will select the row associated with Ted's profile. Using both the statements we can perform the above attack as id is a primary key for the users in database and username is unique key for the users in database. So for each user there will be only one id and username value in database.

The database stores all password's sha1 values in the database. We have calculated the sha1 function value for string 'attack' using online calculator(reference: https://www.tools4noobs.com/online_php_functions/sha1/), so that we can store sha1 value of string 'attack' in database and we can access Ted's profile using 'attack' string as password.

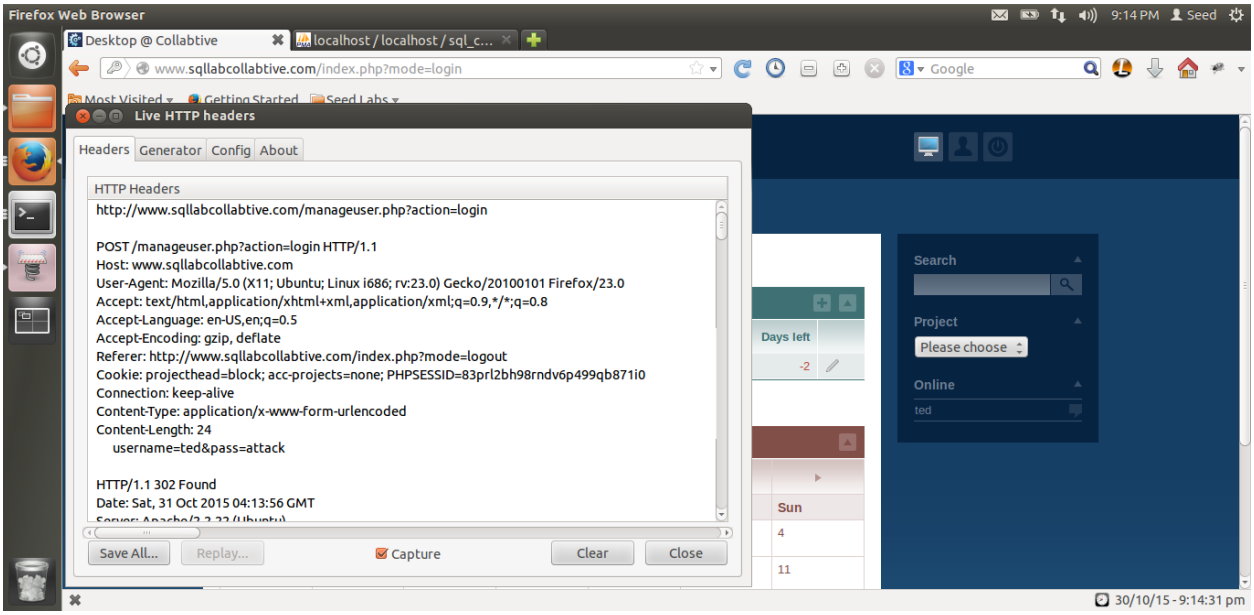
Logging into Ted's Profile:



Now we can confirm that our attack was successful by trying to log into the Ted's profile using 'attack' string as password.

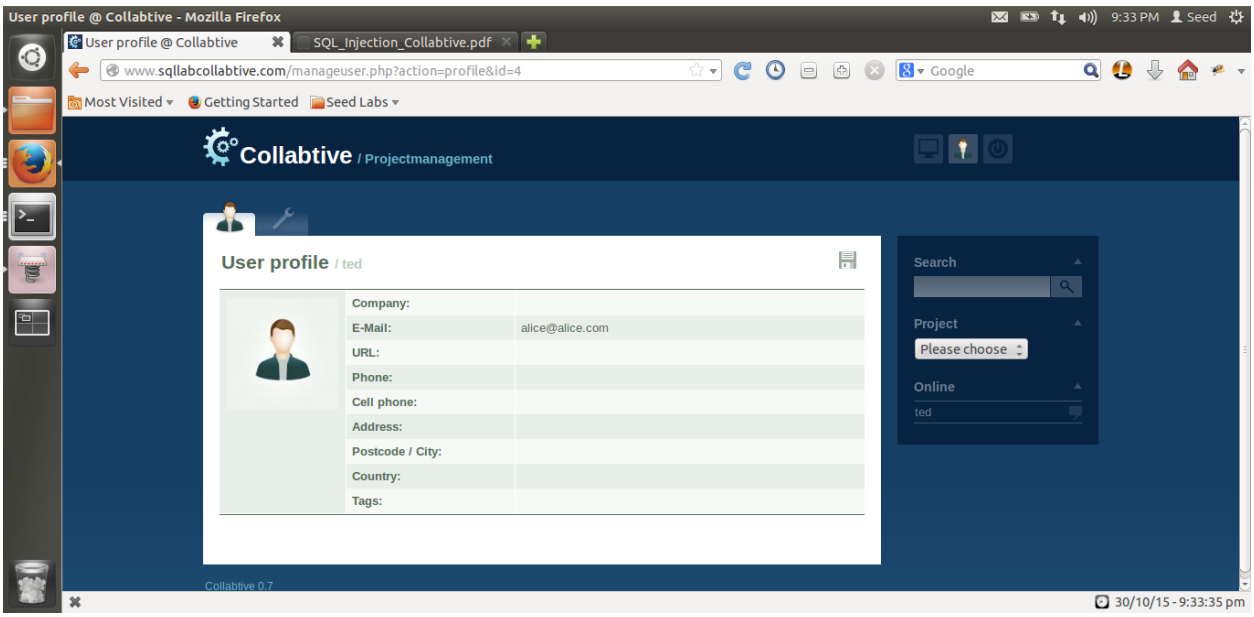
Yogesh Chaudhari
SUID: 244195971

Live HTTP Header login request:



Here we can see that our attack was successful and we were able to log into the Ted's profile using 'attack' string as password.

Ted's Profile: (Modified by SQL Injection attack)



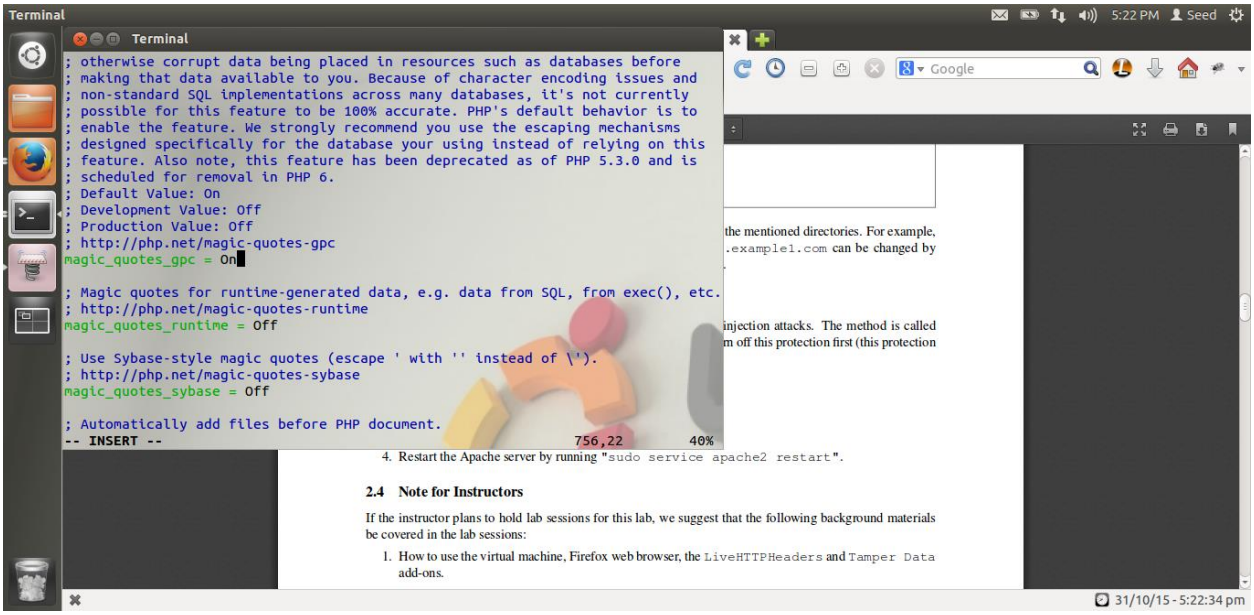
In the above screenshot we can see that Ted's profile was also modified by our SQL injection attack and in email section of Ted's profile we have Alice's email ID.

Yogesh Chaudhari
SUID: 244195971

Countermeasures:

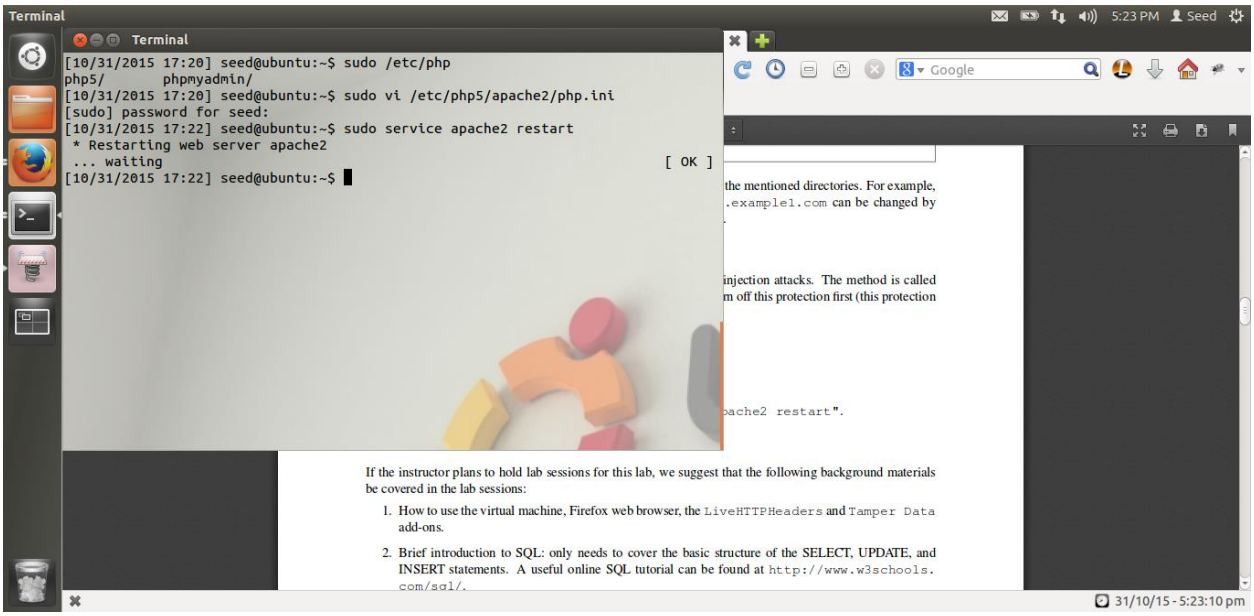
1. Escaping special characters: using magic_quotes_gpc

Turning on countermeasure:



Here we are turning on countermeasure by using **magic_quotes_gpc = on** command.

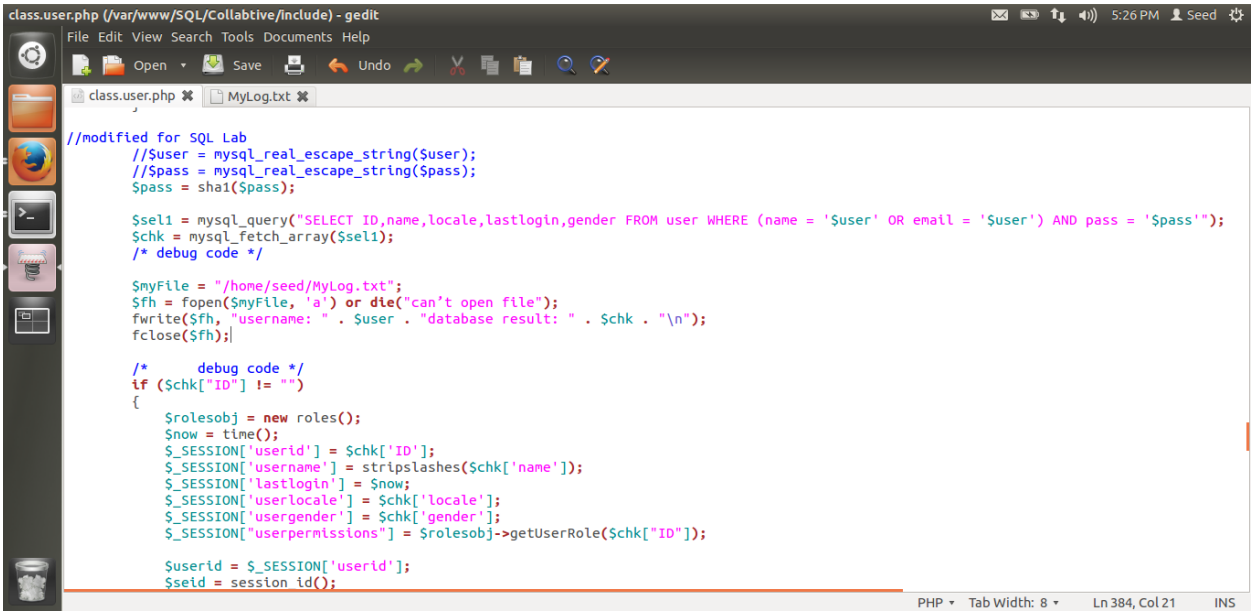
Restarting Apache Server:



Here by using command service apache2 restart, we are restarting our apache server so that the countermeasure effect takes place.

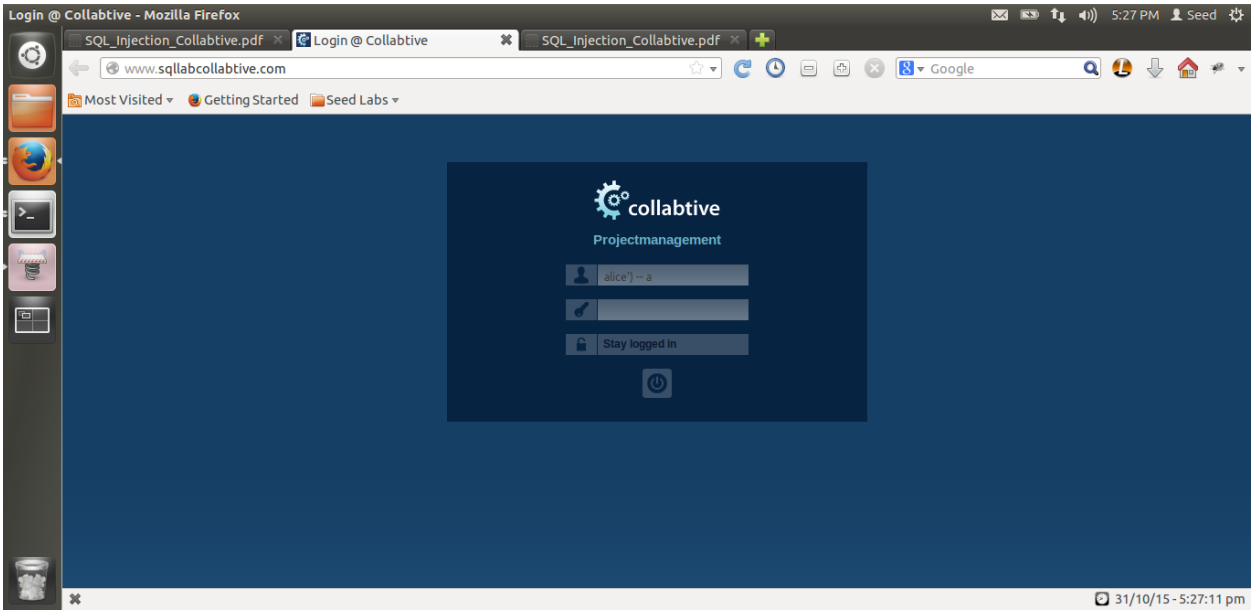
Yogesh Chaudhari
SUID: 244195971

Debug code in class.user.php:



Here we have added code in class.user.php file which will print the php variable file in to external file. We are printing $user php variable in file MyLog.txt (file owned by www-data) to examine how our countermeasure works on server side.

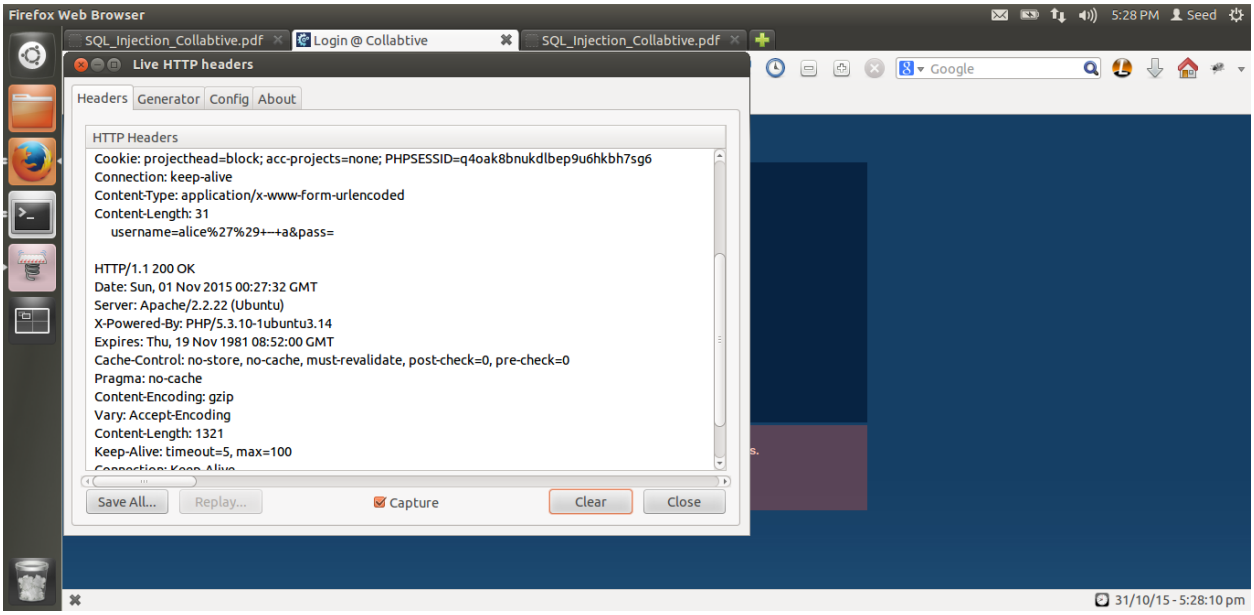
Trying attack:



Here we writing the command as explained in 1st task to try to perform SQL injection attack.

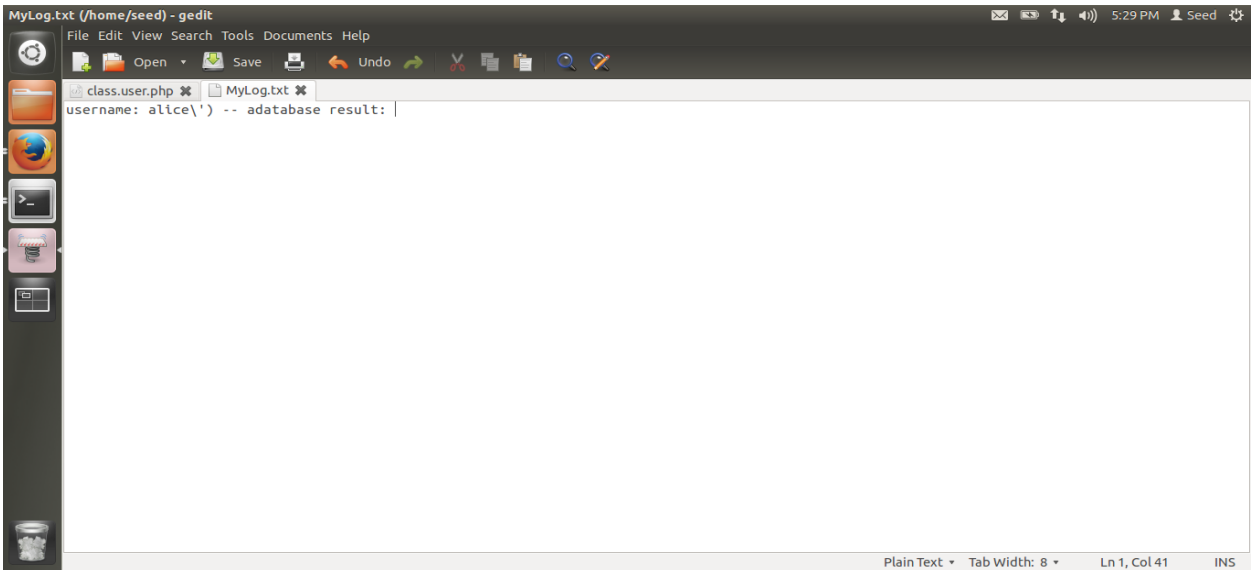
Yogesh Chaudhari
SUID: 244195971

Attack unsuccessful:



From above screenshot we can see that our attack was unsuccessful.

Debug Output:



From the debug output we can see the reason why our attack was unsuccessful. When alice') –a was sent to the server as a user value in request, on server side due to countermeasure magic_quotes_gpc, the server automatically prepended / to special character " ' " and added that as value to $user variable. Hence, this was considered as data and not parsed as code, so our attack was unsuccessful as database was not able to find a user with value "alice') – a" inside its own database.

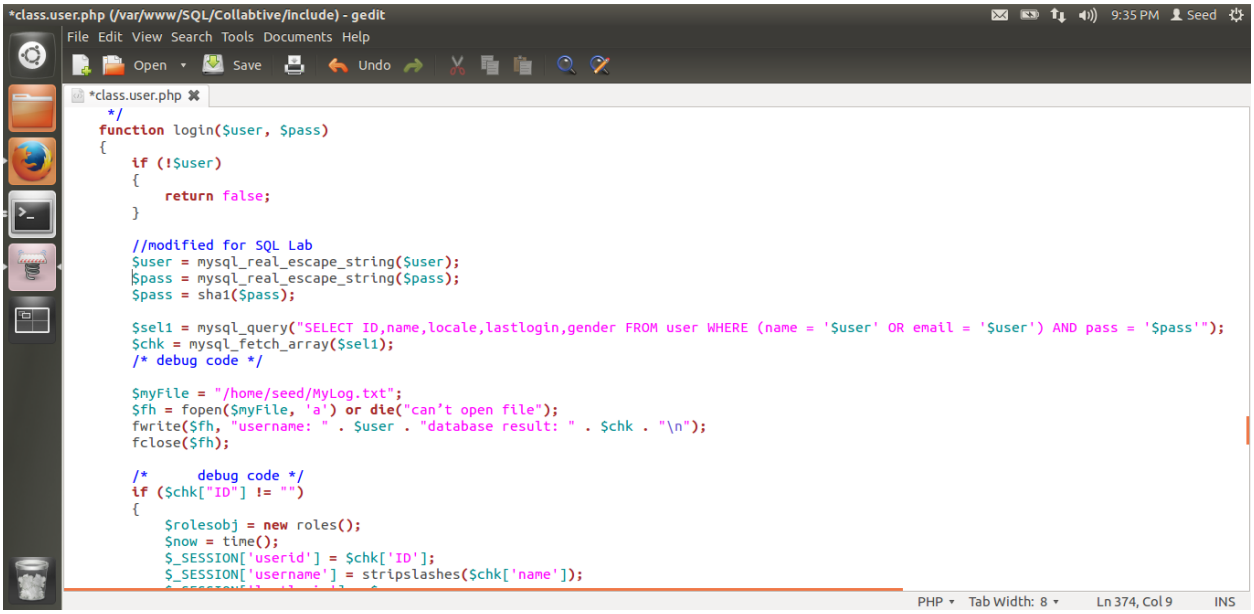
Magic_quotes_gpc() function automatocally prepends '\' character to some special characters when sent through HTTP request as data to the php server. These special characters are ', ", \, NULL.

Yogesh Chaudhari
SUID: 244195971

2. Escaping special characters using mysql_real_escape_string:

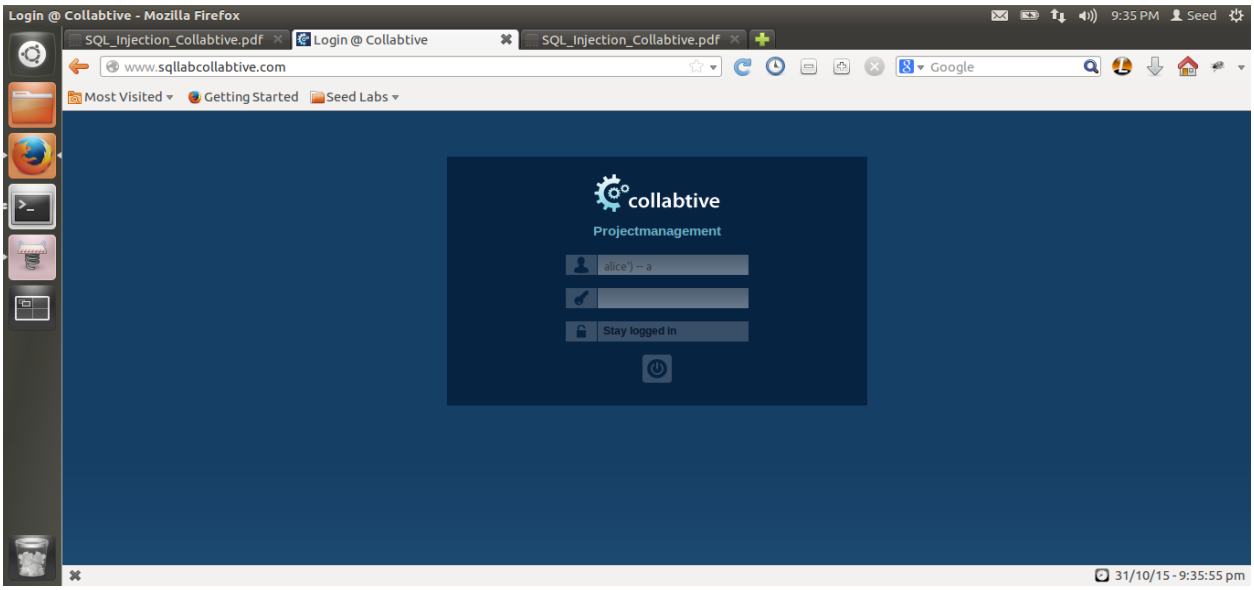
We turned of the magic_quotes_gpc countermeasure by using command **magic_quotes_gpc = off** and then restarted the apache server using command **service apache2 restart.** We have setup debugging code on $user variable as explained in above task.

Turning on mysql_real_escape_string() countermeasure:



Here we are turning on the mysql_real_escape_string() countermeasure on login page by using mysql_real_escape_string() on php variables $user and $pass which stores the HTTP request input values for those input fields.

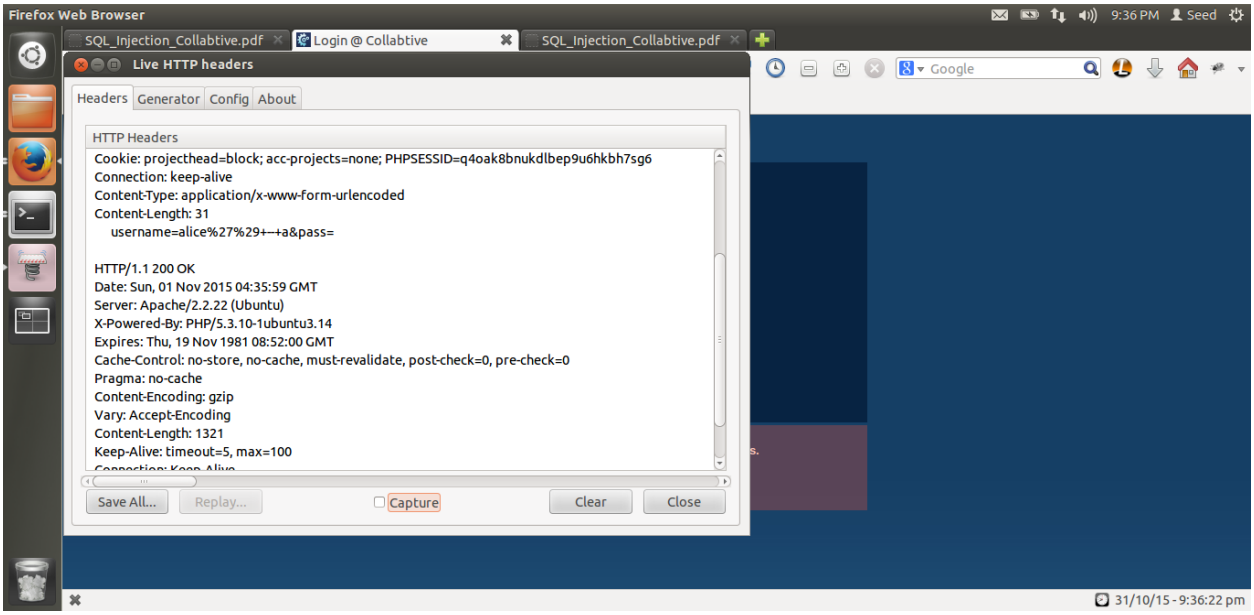
Attack:



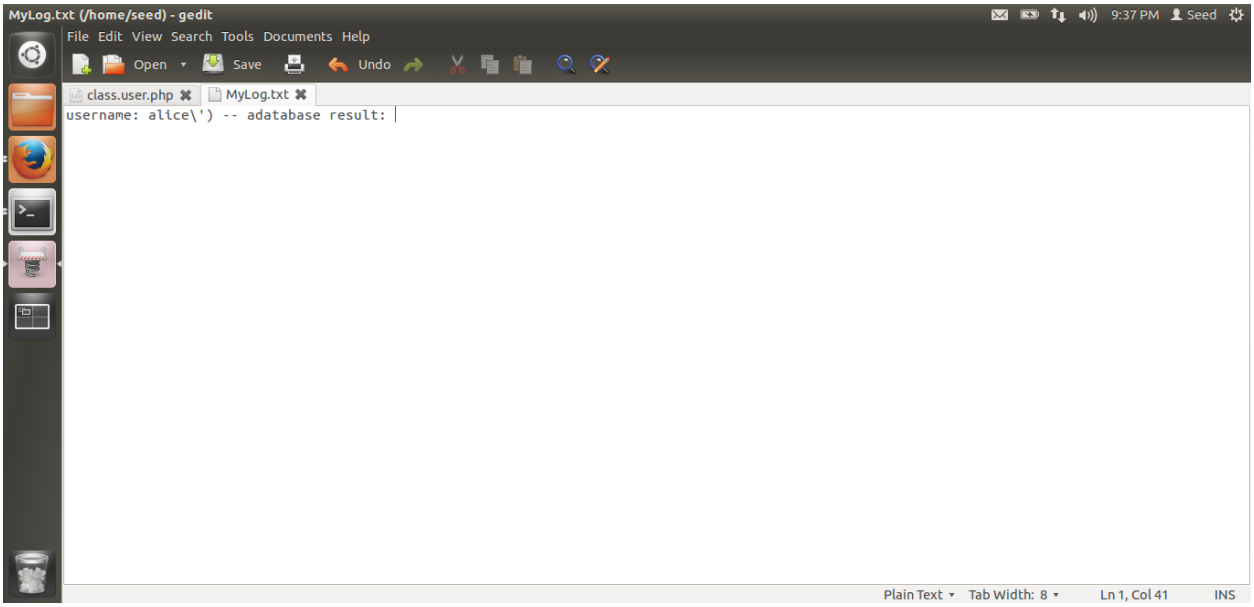
Here we are trying to do SQL injection attack as explained in task 1.

Yogesh Chaudhari
SUID: 244195971

Unsuccessful attack:



From above screenshot we can see that our attack was unsuccessful.



From above screenshot we can see the $user variable value which was substituted in SQL statement. Here, we can see that due to the use of mysql_real_escape_string() function on user variable the / was prepended to special character ‘. Hence, this was considered as data and not parsed as code, so our attack was unsuccessful as database was not able to find a user with value “alice’) – a” inside its own database.

The function mysql_real_escape_String() when used on a string prepends \ to each special character in that string. The special character set depends on the database character set chosen. In normal cases it prepends backslash \ to \x00, \n, \r, \, ', " and \x1a special characters.

Yogesh Chaudhari
SUID: 244195971

3. Using prepare statement:

Code for function login() using prepare statement in class.user.php file:

```
function login($user, $pass)
  {
     if (!$user)
     {
        return false;
     }
     $pass = sha1($pass);

         //code for prepare statement
         $db = new mysqli('localhost', 'root', 'seedubuntu', 'sql_collabtive_db');
         $stmt = $db->prepare("SELECT ID, name, locale, lastlogin, gender FROM user WHERE name=?
AND pass=?");
         $stmt->bind_param("ss", $user, $pass);
         $stmt->execute();
         //The following two functions are only useful for SELECT statements
         $stmt->bind_result($bind_ID, $bind_name, $bind_locale, $bind_lastlogin, $bind_gender);
         $chk=$stmt->fetch();

         if ($bind_ID != "")
     {
        $rolesobj = new roles();
        $now = time();
            $_SESSION['userid'] = $bind_ID;
            $_SESSION['username'] = $bind_name;
            $_SESSION['lastlogin'] = $bind_lastlogin;
```

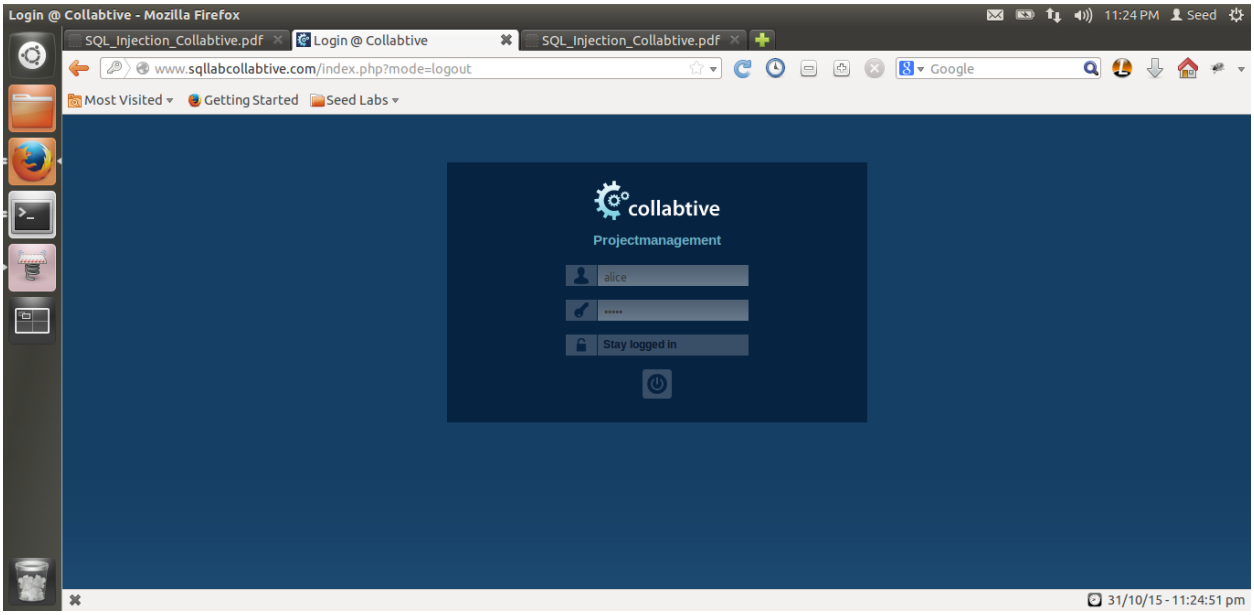
```
            $_SESSION['userlocale'] = $bind_locale;
            $_SESSION['usergender'] = $bind_gender;
            $_SESSION['userpermissions'] = $rolesobj->getUserRole($bind_ID);
        $userid = $_SESSION['userid'];
        $seid = session_id();
        $staylogged = getArrayVal($_POST, 'staylogged');

        if ($staylogged == 1)
        {
           setcookie("PHPSESSID", "$seid", time() + 14 * 24 * 3600);
        }
        $upd1 = mysql_query("UPDATE user SET lastlogin = '$now' WHERE ID = $userid");
        return true;
     }
     else
     {
        return false;
     }
  }
```

Here we are writing the code for using prepare statement to make query on database. For this we need to use my_sqli() function. This requires four arguments, viz., domain or server (in our case localhost), databse name, user name and user password for database mysql. When using prepare statement the server php files sends the query related information for each query twice to the database, first it sends the code to the databse given in prepare() function and then it send the server data values which needs to be substituted or put into the first code sent using bind_param() function. In this function the first part defines the type of data value which is sent to the database by the server. Here, code and data both are sent from server to database separately. Using execute() function the query can be executed on to the database.

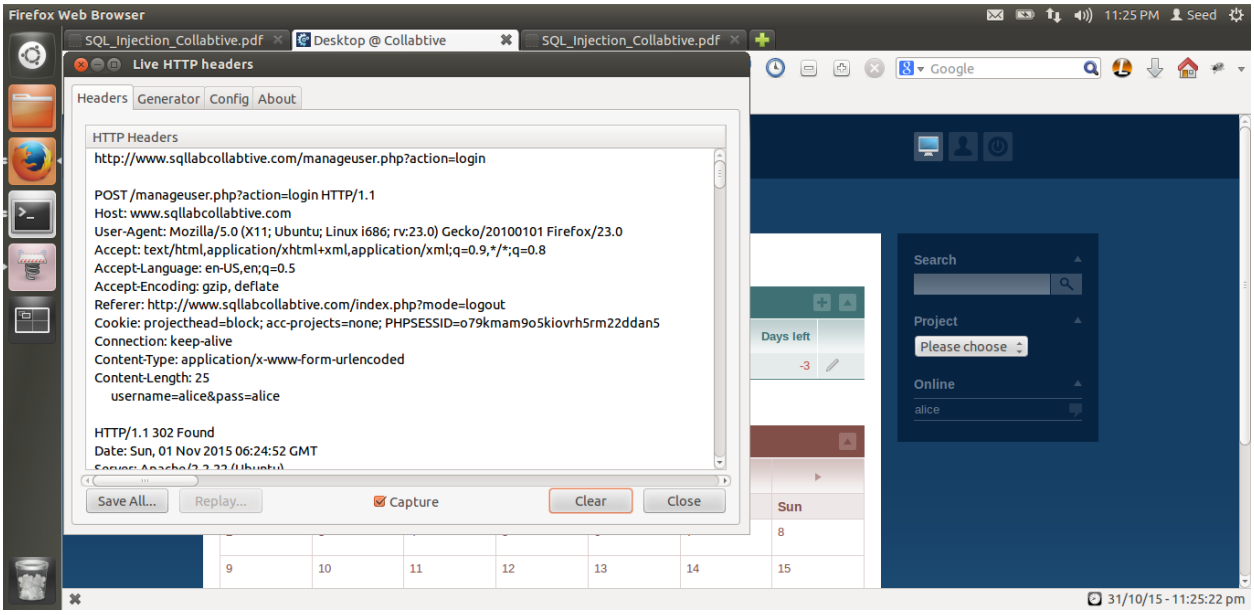
Yogesh Chaudhari
SUID: 244195971

Confirming the working status of the code:



Here we will try to login as Alice into collabtive application to make sure that our prepare statement works perfectly and there are no syntax or any other types of errors.

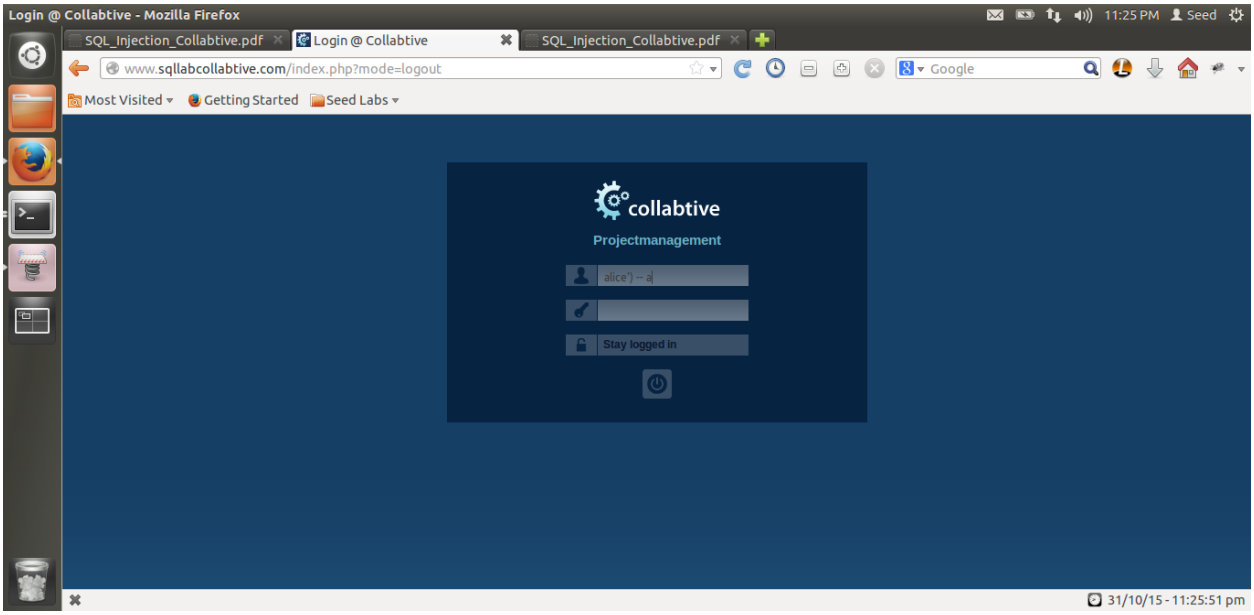
Successful Login:



Here we can see that our prepare statement code runs perfectly without any errors and hence we were able to log into the Alice's profile.

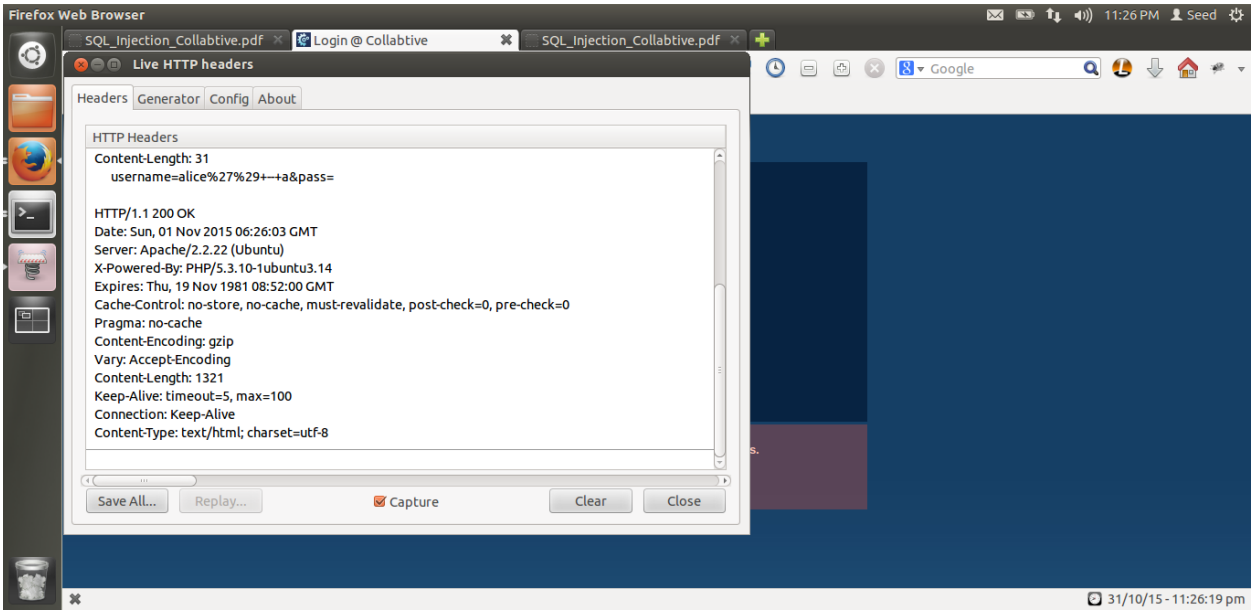
Yogesh Chaudhari
SUID: 244195971

Attack:



Here we will try to perform SQL injection attack on login page of collabtive application by using command alice') – a as explained in the first task.

Unsuccessful attack:



Here we can see that our attack was unsuccessful. Due to the use of prepare statement any code we sent through the input field user to the server, was not parsed at the server side and sent to the database separately from query code. While sending these values there data types are also sent to the

database, hence does not mix the code with the data passed by the user to the server. Hence, data given as input by user doesnot get parsed and is only treated as data value. Hence, when database tries to execute query with the given user variable value in SQL statement, it will look for username which is alice') – a. As thereis no record matching this string the database will send back the null value to the server and hence no record will be selected. Thus, our attack was unsuccessful when prepare() is used.