Escola Tècnica Superior d'Enginyeria Informática

Universitat Politècnica de València

Desarrollo de un plugin de seguridad para WordPress

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Montañés Navarro, Edgar

Tutor: Pons Terol, Julio

2015/2016

Resumen

WordPress® es el CMS más popular en la actualidad y también uno de los mayores objetivos de los hackers. En el TFG se estudiarán los puntos débiles, los plugins de seguridad existentes y se desarrollará un plugin de seguridad que incluya las funcionalidades mínimas que se consideren necesarias y que funcione de forma autónoma, sin necesidad de que el usuario se registre o utilice un servicio externo. Además se plantearán herramientas para servidores Linux de forma que un administrador pueda interactuar fácilmente con la información que genere el plugin.

Palabras clave: CMS, WordPress®, Plugin, Seguridad, Linux.

Abstract

WordPress® is nowadays the most popular CMS and one of the main hacker's objetives. In this EOG, we will study the Wordpress® weak points, the existing security plugins, and we will develop a security plugin that cointains the minimun funcionalities considered necesaries and that works autonomously, without needing a user registration or using an external service. Furthermore, tools will be set out for Linux servers so that a system administrator can interact easily with the information generated by the plugin.

Keywords: CMS, WordPress®, Plugin, Security, Linux.

Tabla de contenidos

Índice de figuras

Índice de cuadros

Índice de tablas

1 Introducción

WordPress® [1] es un sistema de gestión de contenidos que proporciona estética, estándares web y usabilidad; y que destaca por su facilidad de instalación, uso y modificación. En la actualidad, es uno de los gestores de contenidos más utilizados y, uno de los principales usos por el cual es predominante es, por su utilización para la creación de blogs.

Se conoce por sistema de gestión de contenidos o CMS [2](proviene del inglés *Content Management System*) a la plataforma que permite desarrollar una estructura para la creación y administración de contenidos. Esta plataforma consiste en una interfaz que controla una base de datos donde se aloja el contenido. El sistema permite manejar de manera independiente el contenido y el diseño. De esta manera, es posible manejar el contenido y darle un diseño distinto al sitio web, sin tener que darle formato al texto.

En el caso de WordPress® , la plataforma se encuentra diseñada en lenguaje interpretado PHP, y es por eso que, solo necesita de un servidor web y una base de datos para funcionar. Lo más habitual es, que se utilice Apache como servidor web y MySQL como base de datos.

Llegados a este punto, es posible que surjan varias preguntas, tales como: ¿qué es lo que lo hace realmente atrayente? ¿por qué se dice que es un gestor de contenidos fácil de adaptar y modificar? ¿A qué se debe? Respondiendo a estas cuestiones, WordPress® destaca, básicamente, por dos aspectos: admite plantillas y complementos o *plugins*[1]. El uso de plantillas facilita el diseño del contenido, sin tener que preocuparse de formatear el texto, mientras que el uso de *plugins* permite añadir nueva funcionalidad a dicho gestor de manera sencilla, sin necesidad de modificar el código del núcleo de la plataforma manualmente.

Otro aspecto importante a subrayar, y que también lo hace realmente atractivo, es la estructura de directorios que utiliza para almacenar y separar los diferentes tipos de información que no se almacenan en la base de datos. Esto proporciona al usuario un criterio claro de dónde debe dirigirse, dependiendo del tipo de información que esté buscando. Se realiza una muestra de dicha estructura en el cuadro 1.

A continuación se describe brevemente qué tipo de información se almacena en cada directorio:

[1]En la terminología de WordPress®, se utiliza generalmente, el término en inglés *plugin* para denominar a los complementos.

```
user@HOST:/var/www/html/wordpress/wp-content# ls -l
drwxr-xr-x 4 www-data www-data 4096 oct  4 17:12 languages
drwxr-xr-x 8 www-data www-data 4096 oct 17 12:46 plugins
drwxr-xr-x 5 www-data www-data 4096 oct  4 16:05 themes
drwxr-xr-x 2 www-data www-data 4096 oct 17 12:46 upgrade
drwxr-xr-x 3 www-data www-data 4096 oct 13 21:33 uploads
```

Cuadro 1: Estructura de directorios para el almacenamiento de contenido en WordPress®

.

- *languages*: en este directorio se almacenan las traducciones de los diferentes temas y plugins alojados, para que se visualicen en el mismo idioma que la instalación de WordPress® , eso si, siempre y cuando, el desarrollador se haya preocupado en realiarlas, mediante la creación de los archivos correspondientes.

- *plugins*: en este directorio se almacenan todos los *plugins* que se deseen añadir para ampliar la funcionalidad y caraterísticas del gestor de contenidos.

- *themes*: en este directorio se almacenan todos los temas y diferentes plantillas que se deseen añadir al gestor de contenidos. Tal y como se ha comentado ya, la funcionalidad de los temas no es otra que cambiar el diseño, el aspecto, y la visualización de WordPress® .

- *upgrade*: en esta carpeta se almacenan las actualizaciones que se descargan en el servidor, tanto de plantillas y temas, como de complementos o plugins. Posteriormente, cuando se finaliza la actualización, los archivos son borrados automáticamente.

- *uploads*: por último, en este directorio se almacenan todos los archivos multimedia que se suben al servidor mediante el gestor de contenidos y que, normalmente, deseamos que estén disponibles posteriormente, para añadirlos en algún artículo.

1.1. Presentación del problema

El hecho de que WordPress® sea uno de los gestores de contenidos más conocidos y utilizados en la actualidad, implica que también sea uno de los más atacados por los *hackers* en busca de vulnerabilidades. Así pues, para evitar que el sitio web se vea comprometido y, por consecuencia también pueda verse comprometido nuestro servidor web, se ha diseñado un plugin con diversas herramientas que ayudan a que el CMS WordPress® esté protegido

y, además, esta protección sea en tiempo real y sin ayuda de ningún servicio de terceros.

1.2. Objetivos

El objetivo fundamental de este proyecto es crear un plugin que proporcione seguridad al gestor de contenidos WordPress®. Para conseguir este objetivo, se han de alcanzar primero, diversos hitos o subobjetivos que incluyen desde: la instalación y la inicialización del CMS WordPress®, hasta el diseño y pruebas de funcionamiento del plugin. En rasgos generales, este proceso se dividen en las siguientes etapas:

- Instalación del CMS WordPress® y creación de la base de datos correspondiente.
- Estudio de las funciones de WordPress® para el correcto desarrollo del plugin.
- Creación de un plugin sin funcionalidad.
- Añadir funcionalidad al plugin creado.
- Pruebas del correcto funcionamiento del plugin instalándolo en un entorno real.

Una vez se ha diseñado el proceso a seguir para alcanzar el objetivo final, y dada la necesidad de acotar el alcance del proyecto, después de haber realizado el estudio de las diversas funcionalidades que tienen los plugins de seguridad existentes, se han definido funcionalidades concretas y concisas que se desearía que tuviera el nuevo plugin. A continuación se describen las funcionalidades:

- Bloqueo de direcciones IP mediante directivas en el fichero .htaccess. También conocido como lista negra o *Blacklist*.
- Notificación por correo electrónico cuando un usuario administrador inicia sesión en el panel de administración de WordPress® .
- Obtener información de una dirección IP.
- Bloquear una dirección IP cuando se intenta acceder al panel de administración con un determinado usuario.
- Habilitar una lista blanca de direcciones IP, para que no se les bloquee el acceso al panel de administración.

- Restringir el acceso al sistema por XML-RPC. XML-RPC nos permite enviar artículos a la instalación WordPress® remotamente, sin necesidad de acceder al panel de administración. Se han encontrado varias vulnerabildades que podrían permitir a un atacante remoto obtener las credenciales de un usuario válido, de ahí, que se busque restringir el acceso a dicha funcionalidad solo desde direcciones IP conocidas, pertenecientes a usuarios legítimos.

- Bloqueo de peticiones POST que obtienen como respuesta del servidor un error http de tipo 404.

1.3. Resumen de la solución

La solución a los objetivos propuestos se ha ido alcanzando de forma gradual, y gracias a que se diseñaron pautas y objetivos claros, concretos y concisos, dando una visión clara que permitió determinar el porcentaje del trabajo realizado en cada momento.

En primer lugar, se almacena toda la información necesaria, mediante tablas específicas que se añaden junto con el resto de tablas existentes en la base de datos creada durante la instalación del gestor de contenidos. Esto permite que, posteriormente, se acceda a la base de datos y se recopile la información deseada para crear diferentes listas, como por ejemplo: lista de usuarios específicos, es decir, los que no están permitidos para iniciar sesión en el panel de administración; lista de direcciones IP bloqueadas, lista de direcciones IP que no se van a proceder a bloquear aunque incumplan alguna de las condiciones que normalmente produciría su bloqueo, lista de direcciones IP que tienen acceso mediante XML-RPC...

Asimismo, al tener el control de la información almacenada en las tablas de la base de datos, es posible realizar determinadas acciones automáticamente, cuando se cumpla determinada condición, lo que en este caso, deriba en el bloqueo automático de una dirección IP.

En relación a la solución alcanzada para bloquear el acceso a las direcciones IP, se ha implementado añadiendo directivas en el fichero .htaccess del directorio raíz. En el caso de que se deseara volver a otorgar acceso a la dirección IP, se ha de borrar la línea correspondiente en dicho fichero .htaccess. Se ha adoptado la misma solución para permitir el acceso mediante XML-RPC, permitiendo uso específico mediante directiva en dicho fichero.

Por otro lado, para el envío de correos electrónicos, cuando un usuario administrador

inicia sesión en el panel de administración del gestor de contenidos WordPress®, se utiliza la función *mail()* de PHP, que usa el servidor de correo local instalado en el propio servidor para el envío de correos. Debido a esto, se ha configurado un servidor de correo para realizar pruebas y comprobar que el envío de correos funciona correctamente. Esto evita tener que añadir las credenciales de acceso de una cuenta de correo existente para poder realizar el envío de los mismos.

1.4. Estructura de la memoria

La memoria de este proyecto consta de los siguientes apartados:

- Un primer capítulo introductorio, que incluye todos los apartados de la memoría presentados hasta ahora: una introducción, que contiene una descripción del problema, y que deriva en la necesidad de crear un plugin de seguridad para el gestor de contenidos. Posteriormente, se expone el objetivo final del proyecto y, para concluir el apartado, se realiza un resumen de la solución alcanzada.
- Un segundo capítulo en el que se presenta un estado del arte en el que se describe una selección de los diferentes plugins de seguridad existentes para el gestor de contenidos WordPress®, y que fueron útiles para determinar qué funcionalidades debía incluir el nuevo plugin a crear.
- Un tercer capítulo donde se esclarece el desarrollo de todas las funcionalidades que incluye el plugin, y cómo se ha llegado a dicha solución.
- Un cuarto capítulo, donde se expone el plugin en un entorno de pruebas real, para determinar su correcto funcionamiento.
- Y por último, un quinto capítulo, en el cual, se exponen las conclusiones tras haber realizado el proyecto, además de exponer nuevas funcionalidades que se podrían añadir en futuras ediciones o proyectos.

Por otro lado, este trabajo también consta de los siguientes anexos:

- Un primer anexo donde se explica la instalación de las herramientas necesarias para el correcto funcionamiento del CMS WordPress®: principalmente, se divide en la instalación del servidor web Apache y, la instalación de la base de datos MySQL en un servidor GNU Linux. Por último, se comenta las modificaciones que se deben

de realizar en el fichero de configuración para que el gestor de contenidos funcione correctamente.

- Un segundo anexo, en el que se muestra la interfaz del plugin realizado.
- Un tercer anexo, donde se adjunta un script que bloquea direcciones IP en el Firewall automáticmante, y que ayudará a un administrador de sistemas a proteger el servidor.

2 Estado del arte

WordPress® es un gestor de contenidos de libre distribución, en donde cada desarrollador puede aportar su propio trabajo, principalmente: nuevas plantillas y nuevos *plugins* [2]. Actualmente, exsiten multitud de plugins que aportan seguridad al mismo, y es por eso que, después de visualizar brevemente muchos de los plugins hallados en la tienda, se han seleccionado varios de ellos, plugins interesantes que su contenido resulta de gran aportación para el proyecto, y se ha hecho una descripción de las funcionalidades halladas en cada uno de ellos. Finalmente, se han seleccionado para su estudio los siguientes plugins: Wordfence, BulletProof Security, Login-lockdown e iThemes Security.

La descripción de cada funcionalidad se ha realizado de una forma breve, sin entrar en excesivos detalles, dado que el objetivo principal del estudio no es analizar con profundidad el funcionamiento interno de cada plugin, sino conocer las diferentes funcionalidades que poseen.

2.1. Wordfence Security

Cuando se realiza una búsqueda en Internet sobre los mejores plugins de seguridad existentes para el CMS WordPress®, uno de los plugins que más se recomienda instalar es éste, Wordfence Security[4]. Es por eso que, en este apartado se pretende describir las distintas funcionalidades y las diferentes características que posee.

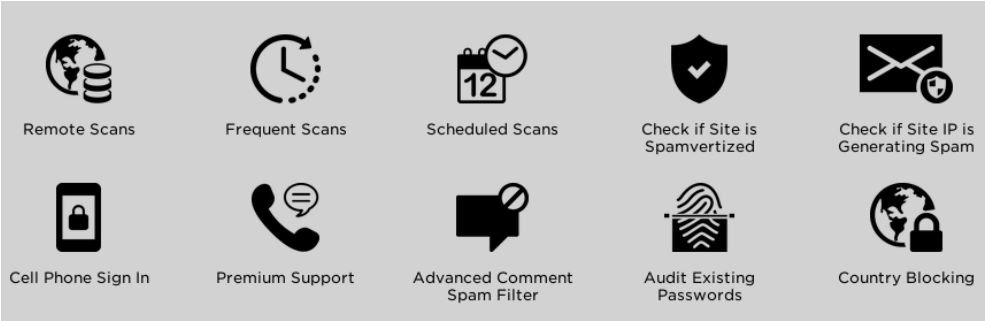

Figura 1: Captura de pantalla que muestra las diferentes opciones de las que consta el plugin WordPress®.[3]

Cuando se accede al panel de administración y se pulsa sobre el nombre del plugin, se

[2]También puede (y debe) reportar errores o vulnerabilidades detectadas.
[3]Imagen obtenida de: [5]

muestran las siguientes opciones:

- Escanear (*Scan*): en este apartado se nos ofrece poder escanear la instalación WordPress® existente en el servidor, en busca de problemas relacionados con la seguridad: se muestra si los archivos poseen los permisos adecuados, si existe alguna divulgación de información...

- Tráfico en tiempo real (*Live traffic*): este plugin es capaz de mostrar en tiempo real todo el tráfico que se está generando en torno a la instalación del WordPress®, de manera que, nos puede ayudar a encontrar usuarios que realizan peticiones malintencionadas, o *crawlers* que pueden hacer descender el rendimiento de nuestro servidor, a causa de los excesos de peticiones a las que debe responder. Esta opción también permite visualizar los usuarios que han conseguido iniciar sesión y acceder al panel de administración.

- Configurar rendimiento (*Perfomance Setup*): en este apartado disponemos de una serie de opciones que nos permiten configurar un sistema de caché, que utiliza Falcon Engine, y que permitirá al servidor atender un mayor número de peticiones en la misma instancia de tiempo. Esto ayudará a que sea más difícil efectuar un ataque de denegación de servicio.

- Bloqueo de direcciones IP (*Blocked IPs*): como su propio nombre indica, en esta sección, se podrá bloquear cualquier dirección IP que se desee. Principalmente, que se detecte que está haciendo un uso excesivo de recursos o que esté intentando comprometer el servidor web. Existe la opción de bloquear una dirección IP temporalmente o permanentemente.

- Auditar contraseñas (*Password Audit*): El propio plugin es capaz de determinar en una escala cuán fácil o difícil de romper es la contraseña de un usuario existente, es decir, cómo de fácil o difícil le resultaría a un atacante remoto obtener la contraseña de un usuario.

- Inicio de sesión móvil (*Cellphone sign-in*): En la versión de pago de este plugin se implementa la posibilidad de autenticarse en el WordPress®en dos pasos. Primero, el usuario se loguea con usuario y contraseña, y si éstos son correctos, se le enviará un mensaje de texto al usuario para que lo añada en su proceso de autenticación. Esto evita que, cualquier atacante pueda conseguir acceso no deseado.

- Bloqueo de acceso por país (*Country Blocking*): La versión de pago de este plugin también incluye la opción de bloquear una dirección IP por geolocalización, ya que hay lugares que son más propensos a realizar ataques.

- Programar escaneo (*Scan schedule*): en la versión de pago se nos permite programar escaneos periódicos automáticos, lo que ayuda a controlar más fácilmente si el WordPress®es vulnerable, debido a que se haya realizado alguna mala configuración, o exista algún plugin desactualizado.

- Whois (*Whois Lookup*): esta es una pequeña herramienta de Whois, que nos facilita la búsqueda de información sobre a quién pertenece, quien es el propietario, y donde está una dirección IP. De esta forma, nos ahorra el tener que utilizar otros medios o herramientas para obtener este tipo de información sobre la dirección IP.

- Bloqueo avanzado (*Advanced Blocking*): esta sección nos ofrece más opciones para el bloqueo de una dirección IP. Así como en el apartado anterior solo podíamos bloquear una dirección IP, en este nuevo apartado se nos permite realizar bloqueo de direcciones IP por rango.

- Opciones de configuración del plugin (*Options*): esta sección está dedicada a ofrecer opciones para poder configurar el propio plugin.

2.2. BulletProof Security

BPS Security [7] es otro de los plugins más famosos existentes relacionados con la securización del gestor de contenidos. Este plugin también tiene unas características que resultaron ser interesantes para su estudio en el momento de realizar este proyecto. Así pues, seguidamente se va a proceder a comentar sus caraterísticas.

Cuando accecemos a las opciones del plugin desde el panel de administración nos encontramos con las siguientes secciones:

- *htaccess Core (Security | Firewalls)*: este plugin contiene un repositorio donde se almacenan ataques maliciosos conocidos. En el caso de que alguna petición coincida con alguna del filtro, ésta es bloqueada. La ventaja que aporta esta forma de trabajar es que no se bloquea a nivel individual, por dirección IP, sino que bloquea cualquier tipo de petición sospechosa.

[4]Imagen obtenida de: [7]

Figura 2: Captura de pantalla que muestra el logo y descripción del plugin BulletProof-Security de WordPress®.[4]

- *Login Security*: te permite registrar todos los inicios de sesión de cuentas de usuarios o registrar sólo intentos de inicios de sesión de usuarios con cuentas bloqueadas. Además, protege contra los intentos de inicio de sesión mediante fuerza bruta. Existe la opción de recibir alertas mediante correo electrónico: recibir alertas de correo electrónico cuando una cuenta de usuario está bloqueada, un administrador inicia sesión... También es posible modificar los mensajes de error de inicio de sesión fallida.

- *Idlse session Logout Cookie Expiration*: Este apartado nos permite configurar el cierre automático de sesión para los usuarios inactivos. Para ello se monitoriza el movimiento de ratón, o pulsaciones de teclado. Además, se nos permite configurar el tiempo de expiración de la cookie de sesión, con lo que nos aseguramos que los usuarios, deban iniciar sesión con su usuario y contraseña cada cierto tiempo y no se quede siempre iniciada en el navegador web.

- *DB Backup*: Permite realizar copias de seguridad de la base de datos, tanto de forma manual, como programar una tarea para que se realicen las copias de manera automática.

- *Security Log*: Por un lado se permite añadir una cuenta de correo electrónico, donde se enviarán ciertas alertas dependiendo de lo que se haya configurado. Por otra parte, se permite añadir user-agent al que se les dará una respuesta por defecto, sin atender la petición solicitada.

- *Maintenance mode*: Opción que nos permite habilitar una página web de mantenimiento con una cuenta regresiva, mientras realizamos cambios en el sitio web. Una

vez esta cuenta atrás finaliza, se vuelve a habilitar automáticamente la respuesta normal y el sitio web vuelve a funcionar con normalidad.

- *System Info*: En esta sección podremos encontrar información interesante del servidor donde está instalado WordPress®, como por ejemplo: versiones de PHP y MySQL, sistema operativo sobre el que está corriendo el CMS, servidor web sobre el que está montado, uso de memoria y la dirección IP asignada al sitio web.

- *UI | UX | Theme Skin | Processing Spinner | ScrollTop Animation | WP Toolbar | Script Style Loader Filter (SLF)*: en este apartado se permite administrar algunas opciones del gestor de contenidos, tales como: cambiar el tema de la interfaz, personalizar la barra de herramientas, etc.

- *Setup Wizard*: Por último, la sección correspondiente donde se permite configurar diferentes opciones correspondientes al plugin.

2.3. Login-lockdown

Este plugin, Login-lockdown[8], tiene una función muy sencilla, pero muy interesante. La única función que tiene es, que registra la dirección IP y la fecha y hora de cada intento de inicio de sesión fallido. En el caso de que hay más de un cierto número de intentos fallidos en un corto período de tiempo desde el mismo rango de direcciones IP, entonces se bloquea el inicio de sesión para todas las peticiones de ese rango de direcciones.

Los valores por defecto del plugin son bloquear 60 min al rango de direcciones IP después de 3 intentos de conexión fallidos en 5 minutos. Estos parámetros se pueden modificar a través del panel de administración.

Figura 3: Captura de pantalla que muestra el logo y descripción del plugin Login LockDown de WordPress®.[5]

El interés de este plugin para el proyecto se debe, principalmente, a que al tener una función tan concreta, el código para programarlo es sencillo. Este hecho ha facilitado el desarrollo del propio plugin, ayudando a entender la estructura y el cómo programarlo.

2.4. iThemes Security

Otro de los plugins que resultan interesantes a la hora de realizar este proyecto es este, iThemes Security. Anteriormente se le conocía como: Better WP Security.

Si leemos en su página web [9], vemos que se define a si mismo como un plugin que evita que el CMS sea vulnerable a ataques automatizados. También refuerza las credenciales de usuario.

Figura 4: Captura de pantalla que muestra el logo y descripción del plugin iThemes Security de WordPress®.[6]

Si entramos en el panel de administración del CMS vemos que este plugin consta de las siguientes opciones:

- *Dashboard*: En esta sección se presentan las vulnerabilidades encontradas en la instalación WordPress®. Las debilidades hayadas se presentan por orden, de mayor a menor gravedad.

- *Settings*: en esta sección se permite configurar opciones interesantes, tales como: notificaciones por email, bloqueo de una dirección IP por excesivos intentos de inicio de sesión fallidos, tipos de logs que se desean almacenar en el servidor, durante cuánto tiempo y la ruta de los archivos de log...

[5] Imagen obtenida de: [8]
[6] Imagen obtenida de: [9]

- *Advanced*: ofrece medidas de seguridad para el gestor de contenidos. En este caso, permite modificar parámetros en la configuración, que dificultan que los ataques recibidos sean efectivos. Permite, entre otras cosas: cambiar prefijos de la base de datos, cambiar rutas de directorios conocidos...

- *Backups*: En este apartado se ofrece una estrategia de backup (copia de respaldo), es decir, se recomienda una serie de acciones y medidas que se deben tomar en relación a las copias de seguridad. Por así decirlo, se ofrece una polícita a seguir: Se debe ser consciente de que con solo una copia de seguridad de la base de datos no es suficiente, y que se deben realizar copias de seguridad de archivos y contenido. Además, se debe programar una tarea automática para que estas copias se realicen de manera automática de manera continuada.

- *Logs*: Esta sección es un visor de archivos de log. Facilita la visualización de los mismos, sin necesidad de tener que ir a la ruta física del archivo y el tener que visualizarlo con un software que esté instalado en el equipo o servidor.

- *Help*: Se ofrecen enlaces a internet dónde podrás encontrar soporte en caso de que exista algún problema. Además de obtener respuestas a las dudas o preguntas que puedan surgir.

- *Go Pro*: Esta sección es un enlace directo a la web, en caso de que se deseen aumentar las características ofrecidas por la versión gratuita del plugin.

3 Desarrollo del Plugin

Tanto si se desea crear un tema nuevo, como si se desea crear un plugin, lo primero que se debe de hacer es, crear su directorio correspondiente. En el caso de que sea un tema, la ruta donde se deberá crear el directorio será: *directorio_wordpress/wp-content/themes/*, y en caso de que se desee crear un plugin, la ruta será: *directorio_wordpress/wp-content/plugins/*. Tal y como se cita en la página 12 del libro *Professional WordPress Plugin Development*[10], es una buena práctica nombrar el plugin basándose en la función que va a desempeñar.

En este caso, dado que lo que se desea diseñar es un plugin de seguridad para WordPress®, se crea la carpeta *wp-security-plugin*, que consecuentemente con lo descrito, tendrá la ruta: */directorio_wordpress/wp-content/plugins/wp-security-plugin/*.

Una vez creado el directorio, se debe crear un fichero PHP, donde se comenzará a añadir el código deseado. Dicho fichero debe cumplir con ciertos requisitos básicos para ser interpretado por el núcleo de la instalación WordPress®. Estos requisitos se traducen en que se deben incluir en el fichero diversas líneas de código, como por ejemplo: nombre del plugin, descripción, autor, descripción del tipo de licencia... A continuación se realiza una muestra de lo descrito:

```
1  <?php
2  /**
3  Plugin Name: WP Security Plugin
4  Description: Simply security plugin for WordPress.
5  Version: 1.0.0
6  Author: Edgar Montanes Navarro
7  Author URI:
8  License: GPLv2
9  */
10
11 ?>
```

Cuadro 2: Código PHP básico para que el plugin sea interpretado por la instalación.

Una vez que se han añadido estas líneas de código, si accedemos a la sección de plugins de la instalación WordPress®se visualizará una nueva entrada, tal y como se muesta en la

figura 5. Se ha de remarcar que, por el momento, todas las acciones realizadas hasta ahora, han sido para la creación del plugin, pero que todavía éste, no tiene ninguna funcionalidad, es decir, este plugin no realiza ninguna acción, y que para que esto ocurra, se deben diseñar nuevas funciones, que se traducirá en añadir nuevas líneas de código.

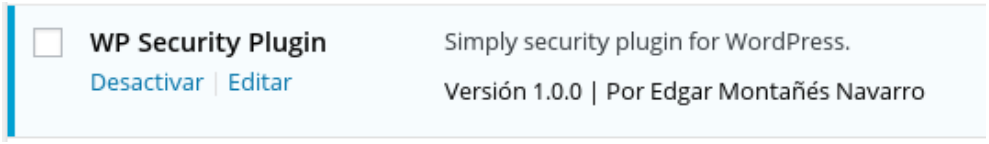

Figura 5: Captura de pantalla que muestra el nuevo plugin creado. [7]

Otra buena práctica, que se menciona también en dicho libro, y que se debe realizar justo después de añadir las cabeceras que se acaban de mostrar, es que justo debajo, se añada la licencia correspondiente para el plugin. En este caso, como se ha mostrado, se ha escogido el tipo de licencia GPLv2, la cual permite la libertad de usar, estudiar, compartir (copiar) y modificar el software [26]. Así pues, se ha añadido esta licencia modificando los campos necesarios, tales como el año de creación y el nombre del autor. Paralelamente, se crea un fichero con nombre: license.txt, y se adjunta la misma licencia nuevamente.

Una vez, después de haber realizado estos primeros pasos, ya se puede empezar a programar las diferentes funciones PHP que otorgarán funcionalidad al plugin. En este caso, antes de comenzar con la programación, se ha decidido añadir una breve guía explicativa de instalación, que describe en tres sencillos pasos qué se debe hacer para activar y poder utilizar el plugin en la instalación WordPress®.

Las dos primeras funciones que se deben programar, nada más comenzar el diseño del plugin, ya que son de carácter imprescindible, son las funciones de activación y desactivación del mismo.

3.1. Función de activación

En esta función se deben contemplar todas aquellas funcionalidades que son necesarias internamente para el correcto funcionamiento del plugin.

[7] Imagen obtenida de aplicar un recorte en el panel de administración de la instalación WordPress® que se ejecuta en el servidor.

3.1.1. Creación de tablas en la base de datos

En este caso, dado que lo que se desea es mantener almacenadas listas de direcciones IP y usuarios, entre otras cosas, necesitamos almacenar información, y además, necesitamos que la información almacenada sea persistente. Así pues, lo que se hace es diseñar varias tablas en la base datos utilizada por la instalación WordPress®.

Por un lado, se crea una tabla, llamada *wp_securityplugin*, en la que se almacenarán todas aquellas direcciones IP que no se desea que dispongan de acceso a la instalación WordPress®, o lo más lógico en un entorno real, al dominio alojado en el servidor web.

Además, se desean mantener listas persistentes de usuarios que provocarán el bloqueo de la dirección IP de forma inmediata, cuando se intente iniciar sesión en el panel de administración con alguno de los usuarios que esté catalogado en dicha lista. Paralelamente, se desea mantener almacenada una lista de direcciones IP que no serán bloqueadas nunca. A la lista que se describe al principio de este párrafo se le conoce con el término de lista negra, del inglés *Blacklist*, mientras que, a la lista descrita posteriormente se le conoce como lista blanca, del inglés *Whitelist*. Para mantener almacenadas ambas listas, se crearán dos tablas nuevas en la base de datos, que tendrán los nombres *wp_securitypluginUserBl* y *wp_securitypluginIPWl*, respectivamente.

Para finalizar con la creación de tablas en la base de datos, se debe crear otra tabla para mantener almacenada una lista de los usuarios que tienen acceso a la instalación WordPress®mediante XML-RPC.

3.1.1.1. Creación de la tabla *wp_securityplugin*

En este apartado se describe cómo está formada la tabla *wp_securityplugin* en la base de datos. La tabla está formada por diversos campos, que se muestran a continuación:

- *id:* de tipo entero, nos indica el orden en que se almacena las direcciones IP en la tabla. Este valor es autoincremental y también es la clave primaria de la tabla.
- *IP:* En este campo de la tabla se almacena la dirección IP que se desea bloquear. Se guarda en formato texto y en código se realizan comprobaciones antes de almacenar la dirección IP en la base de datos para comprobar que es una dirección IP es válida.
- *Attept:* Se utiliza para contabilizar el número de intentos fallidos que realiza una dirección IP cuando intenta acceder al panel de administración. Al tercer intento

fallido se le bloqueará el acceso total al sitio web.

- *block_date:* Muestra la fecha en que se bloqueó la dirección IP, o el último intento de acceso fallido.
- *blocked_manually:* Esta casilla nos indica si la dirección IP bloqueada se ha bloqueado automáticamente, o si por le contrario, ha sido bloqueada de forma manual. El valor si ha sido bloqueada automáticamente será 0, y en caso de que haya sido bloqueada de forma manual, será 1.

A continuación se muestra una descripción de la tabla. La tabla que se muestra es similar a la que se mostraría al ejecutar el comando *desc* en mysql:

Field	Type	Null	Key	Default	Extra
id	mediumint(9)	NO	PRI	NULL	auto_increment
ip	tinytext	NO		NULL	
attempt	mediumint(9)	NO		NULL	
block_date	datetime	NO		0000-00-00 00:00:00	
blocked_manually	tinyint(1)	NO		NULL	

Tabla 1: Composición de la tabla *wp_securityplugin*.

Una vez ya se conoce cómo está formada dicha tabla, se procede a exponer un ejemplo visual en forma de tabla de cómo se almacena una dirección IP, con la intención de aclarar las posibles dudas que puedan quedar todavía por resolver:

id	ip	attempt	block_date	blocked_manually
1	192.168.0.201	3	2016-01-17 15:59:29	0
2	192.168.0.202	3	2016-01-17 16:01:03	1

Tabla 2: Muestra de dos direcciones IP almacenadas en la tabla *wp_securityplugin*.

3.1.1.2. Diseño de la tabla *wp_securitypluginUserBl*

En esta tabla se desea almacenar una lista de usuarios, que tal y como ya se ha mencionado anteriormente, si se intenta iniciar sesión con alguno de ellos en el panel de administración, provocará que se añada la dirección IP en el fichero *.htaccess*, y por tanto, se niega el acceso total al sitio web.

Dado que solo se desea almacenar esa información, la tabla resultante es más sencilla:

- *idUser:* de tipo entero, nos indica el orden en que se almacenan los usuarios en la tabla. Este valor es autoincremental y también es la clave primaria de la tabla.
- *user:* de tipo texto. En este caso concreto, almacena el nombre de usuario.

A continuación se muestra la tabla con sus correspondientes campos descritos:

Field	Type	Null	Key	Default	Extra
idUser	mediumint(9)	NO	PRI	NULL	auto_increment
user	text	NO		NULL	

Tabla 3: Composición de la tabla *wp_securitypluginUserBl*.

Un ejemplo de cómo se almacenan los usuarios en la base de datos, es el siguiente:

idUser	user
1	admin
2	Administrador

Tabla 4: Muestra de dos usuarios añadidos en la tabla *wp_securitypluginUserBl*.

3.1.1.3. Diseño de la tabla *wp_securitypluginIPWl*

Esta sección describe la tabla de la base de datos que almacena una lista de direcciones IP que no se van a añadir nunca al fichero *.htaccess*. Se podría pensar que esta tabla sería muy similar a la tabla *wp_securityplugin*, sin embargo, esto no ocurre así y la tabla *wp_securitypluginIPWl* es mucho más simple. El porqué es muy sencillo: solo se necesita almacenar la dirección IP, y no se necesita ni almacenar la fecha de inclusión ni ningún otro dato. Así pues, la tabla consta de los siguientes campos:

- *idIP:* de tipo entero, nos indica el orden en que se almacenan las direcciones IP en la tabla. Este valor es autoincremental y también es la clave primaria de la tabla.
- *ipwl:* en este campo se almacena la dirección IP que no se va a bloquear nunca, aunque incumpla las reglas establecidas de bloqueo.

Field	Type	Null	Key	Default	Extra
idIP	mediumint(9)	NO	PRI	NULL	auto_increment
ipwl	tinytext	NO		NULL	

Tabla 5: Composición de la tabla *wp_securitypluginIPWl*.

Una vez se conoce la composición de la tabla, se muestra un ejemplo de cómo quedan almacenadas las direcciones IP en la base de datos:

idIP	ipwl
1	192.168.88.254

Tabla 6: Muestra de una dirección IP añadida en la tabla *wp_securityplugiIPWl*.

3.1.1.4. Diseño de la tabla *wp_securitypluginXMLRPC*

Para finalizar con la creación de tablas en la base de datos, se debe de crear otra tabla, cuyo nombre es: *wp_securitypluginXMLRPC*, y que es muy similar a la tabla ya creada *wp_securitypluginIPWl*. Esta nueva tabla también almacena direcciones IP, con la diferencia de que, la información que nos interesa guardar es otra: a las direcciones IP que se almacenan en esta nueva tabla, se les concede acceso al gestor de contenidos mediante XML-RPC. Consecuentemente con lo anterior citado, esta tabla constará también de dos campos:

- *idIP:* de tipo entero, nos indica el orden en que se almacenan las direcciones IP en la tabla. Este valor es autoincremental y también es la clave primaria de la tabla.
- *ipXMLRPC:* en este campo se almacena la dirección IP a la que se va a permitir acceso mediante XML-RPC.

Field	Type	Null	Key	Default	Extra
idIP	mediumint(9)	NO	PRI	NULL	auto_increment
ipXMLRPC	tinytext	NO		NULL	

Tabla 7: Composición de la tabla *wp_securitypluginXMLRPC*.

En este caso, no se va a poner un ejemplo de como se almacena la información en dicha tabla, ya que se almacena de forma a ideńtica a la mostrada ya en la tabla 6.

3.1.1.5. Código generado para la creación de las tablas

Una vez ya se han expuesto las tablas necesarias para almacenar la información y se ha mostrado un ejemplo de cómo se almacena dicha información en cada tabla, se procede a presentar el código realizado para la creación de las mismas:

```
1    global $wpdb; //
2    //creating wp_securityplugin table
3    $table_name= $wpdb->prefix . "securityplugin";
4    if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") != $table_name ){
5                $sql = "CREATE TABLE " . $table_name . " (
6    id mediumint( 9 ) NOT NULL AUTO_INCREMENT ,
7    ip tinytext NOT NULL ,
8    attempt mediumint( 9 ) NOT NULL ,
9    block_date  datetime NOT NULL default '0000-00-00 00:00:00' ,
10   blocked_manually boolean NOT NULL ,
11   PRIMARY KEY ( `id` )
12 );";
13   }
14   $wpdb->query($sql);
15   //creating wp_securitypluginUserBl table
16   $table_name = $wpdb->prefix . "securitypluginUserBl";
17 ...
```

```
...
  if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") != $table_name ){
      $sql = "CREATE TABLE " . $table_name . " (
    idUser mediumint( 9 ) NOT NULL AUTO_INCREMENT ,
    user text NOT NULL ,
    PRIMARY KEY ( `idUser` )
  );";
  }
  $wpdb->query($sql);
    //creating wp_securitypluginIPWl table
  $table_name = $wpdb->prefix . "securitypluginIPWl";
  if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") != $table_name ){
      $sql = "CREATE TABLE " . $table_name . " (
    idIP mediumint( 9 ) NOT NULL AUTO_INCREMENT ,
    ipWl tinytext NOT NULL ,
    PRIMARY KEY ( `idIP` )
  );";
  }
  $wpdb->query($sql);
  //creating wp_securitypluginXMLRPC table
  $table_name = $wpdb->prefix . "securitypluginXMLRPC";
  if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") != $table_name ){
      $sql = "CREATE TABLE " . $table_name . " (
    idIP mediumint( 9 ) NOT NULL AUTO_INCREMENT ,
    ipXMLRPC tinytext NOT NULL ,
    PRIMARY KEY ( `idIP` )
  );";
  }
  $wpdb->query($sql);
```

Cuadro 3: Código PHP realizado para crear las tablas en la base de datos.

3.1.2. Cambios en el fichero *.htaccess*

Hoy en día, es habitual compartir recursos y servidor con otros usuarios, y que el servidor posea varios dominios alojados. Por lo tanto, es muy posible que no se tengan privilegios de administrador sobre el sistema de la máquina, y eso implica que, no tengamos los suficientes privilegios como para realizar cambios directamente en las reglas del *Firewall* (Cortafuegos) del sistema. Así pues, se ha de buscar otra forma de bloquear o restringir el

acceso de una dirección IP al sitio web. Una de las formas con las cuales es posible lograrlo es realizando cambios en el fichero *.htaccess* que se encuentra en el directorio raíz de la instalación WordPress® o del dominio.

La mayoría de entornos web soportan los cambios que se realizan en el fichero *.htaccess*, sin embargo, en el servidor utilizado para realizar las pruebas, se han tenido que realizar cambios en la configuración del servidor web para que las líneas de tipo: *Deny from* o *Allow from* tuvieran efecto. Dichos cambios se muestran a continuación. En la parte de la izquierda se muestra la configuración del servidor original, mientras que en la derecha, se muestra la configuración con los cambios pertinentes:

```
1 //Fichero apache.conf original.                  //Fichero apache.conf modificado
2 <Directory /var/www/>                            <Directory /var/www/>
3   Options Indexes FollowSymLinks                     Options Indexes FollowSymLinks
4   AllowOverride None                           |     AllowOverride All
5                                                >     Order allow,deny
6                                                >     Allow from all
7   Require all granted                                Require all granted
8 </Directory>                                     </Directory>
```

Cuadro 4: Diferencias entre el fichero *apache.conf* original y el modificado.

3.1.2.1. Bloquear el acceso de una dirección IP

En este caso, para bloquear el acceso al sitio web de una dirección IP se crea una sección específica en el fichero *.htaccess*. Eso se traduce en que, se añaden una serie de líneas características en dicho fichero al activar el plugin, que posteriormente, nos ayudan a realizar un correcto manejo del mismo.

Así pues, otra de las funciones que realiza la funcioń de activación, es escribir las siguientes líneas en dicho fichero:

```
1 ###WP-Secuirty-Pl###
2 #Blocking IP address
3 Order deny,allow
4 ###End Of WP-Security plugin
```

Cuadro 5: Muestra de las líneas que se añaden en el fichero *.htaccess* cuando se pulsa la función de activación.

De esta manera, cuando no se quiera permitir el acceso al sitio web de una dirección IP, y que reciba un error http de tipo: *403 Forbidden* cuando intente acceder al dominio, lo único que deberemos hacer es añadir una línea, tal y como se muestra a continuación:

```
Deny from $IP
```

Cuadro 6: Muestra de la línea que se añade en el fichero *.htaccess* para bloquear el acceso a una dirección IP.

Siendo $IP la dirección IP a bloquear, justo antes de llegar a la línea que marca la finalización de escritura en este plugin. Una vez se hayan realizado estos cambios, la dirección IP añadida, no podrá acceder al dominio, evitando así que pueda iniciar sesión o explotar cualquier vulnerabilidad existente.

3.1.2.2. Restringir el acceso mediante XML-RPC

De la misma manera, para poder restringir el acceso mediante XML-RPC, es necesario escribir nuevas líneas en el archivo *.htaccess*. En este caso, el funcionamiento es justo el contrario que el aplicado anteriormente: en el caso anterior, se permite el acceso al dominio a todas las direcciones IP y se va denegando el acceso una a una, mientras que en este caso, se niega el acceso mediante XML-RPC a todas las direcciones IP, y se va habilitando una a una. El código que se debe añadir es el siguiente:

```
# Block WordPress xmlrpc.php requests
<Files xmlrpc.php>
Order deny,allow
Deny from all
</Files>
```

Cuadro 7: Líneas añadidas para denegar el acceso mediante XML-RPC.

En este caso, cuando deseemos que una dirección IP tenga acceso, debemos añadir la siguiente línea:

```
Allow from $IP
```

Cuadro 8: Línea añadida para permitir el acceso a una dirección IP mediante XML-RPC.

Siendo $IP la dirección IP a la que se desea conceder el acceso.

3.1.2.3. Código generado para la añadir las líneas al fichero *.htaccess*

El código que se ha realizado para poder añadir las líneas pertinentes en el archivo *.htaccess* es el siguiente:

```
1  //Adding lines in .htaccess files
2  define (ROOT_DIR, get_home_path() );
3  $file = ROOT_DIR . '/.htaccess';
4  //.htaccess exists?
5  if (! file_exists($file)) {
6    touch($file);
7  }
8  $actual = @fopen(ROOT_DIR . '/.htaccess', 'r' );
9  $esta = ""; $estaXMLRPC = "";
10 while(!feof($actual)) {
11   $linea = fgets($actual);
12   if ($linea === "<Files xmlrpc.php>\n") $estaXMLRPC = true;
13   if ($linea === "###WP-Secuirty-Pl###\n") $esta = true;
14 }
15 if (!$estaXMLRPC) {
16   $actual = @file_get_contents(ROOT_DIR . '/.htaccess');
17   $f = @fopen(ROOT_DIR . '/.htaccess', 'w+' );
18   @fwrite($f, $actual . "# Block WordPress xmlrpc.php requests\n<Files
     xmlrpc.php>\nOrder deny,allow\nDeny from all\n</Files>\n\n");
19   @fclose($f);
20 }
21 if (!$esta) {
22   $actual = @file_get_contents(ROOT_DIR . '/.htaccess');
23   $f = @fopen(ROOT_DIR . '/.htaccess', 'w+' );
24   @fwrite($f, $actual . "###WP-Secuirty-Pl###\n#Blocking IP address\
     nOrder deny,allow\n###End Of WP-Security plugin\n\n");
25   @fclose($f);
26 }
27 @fclose($actual);
```

Cuadro 9: Código PHP realizado para añadir las líneas en el fichero *.htaccess*.

3.2. Función de desactivación

El cometido de esta función es deshacer los cambios ocasionados por la función de activación del plugin en la instalación WordPress®. En este caso concreto, se deben eliminar la tablas creadas en la base de datos y, además, se debe retornar el fichero *.htaccess* al estado original o anterior, antes de que se hubiera realizado la instalación del plugin.

3.2.1. Borrado de tablas

Las tablas que se deben eliminar en la base de datos son las siguientes: *wp_securityplugin*, *wp_securityplugin*$UserBl$, *wp_securityplugin*$User$, *wp_securityplugin*$IPWl$ y *wp_securityplugin-XMLRPC*.

3.2.1.1. Código realizado para la eliminación de las tablas

El código realizado para la eliminación de las tablas en la base de datos ha sido el siguiente:

```
1  global $wpdb;
2  $table_name = $wpdb->prefix . "securityplugin";
3  if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") == $table_name ) {
4    $sql = "DROP TABLE $table_name";
5  }
6  $wpdb->query($sql);
7  $table_name = $wpdb->prefix . "securitypluginUserBl";
8  if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") == $table_name ) {
9    $sql = "DROP TABLE $table_name";
10 }
11 $wpdb->query($sql);
12
13 $table_name = $wpdb->prefix . "securitypluginIPWl";
14 if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") == $table_name ) {
15   $sql = "DROP TABLE $table_name";
16 }
17 $wpdb->query($sql);
18 ...
```

```
17 ...
18   $table_name = $wpdb->prefix . "securitypluginXMLRPC";
19   if( $wpdb->get_var("SHOW TABLES LIKE '$table_name'") == $table_name ) {
20     $sql = "DROP TABLE $table_name";
21   }
22   $wpdb->query($sql);
```

Cuadro 10: Código PHP realizado para eliminar las tablas de la base de datos.

3.2.2. Borrado de líneas en el fichero *.htaccess*

Las líneas que se han de eliminar del fichero *.htaccess* son las añadidas en los cuadros 5 y 7, que son las añadidas para el bloqueo de direcciones IP y las añadidas para restringir el acceso mediante XML-RPC.

3.2.2.1. Código realizado para la eliminación de líneas en el archivo *.htaccess*

En este caso, para la eliminación de las líneas correspondientes en el fichero *.htaccess*, se han creado dos funciones auxiliares similares entre si: una que elimina las líneas correspondientes con el bloqueo de direcciones IP, y la otra función, que elimina todas las líneas relacionado con el acceso mediante XML-RPC.

```
1  function remove_all_related_to_block_IP () {
2    define (ROOT_DIR, get_home_path() );
3    $actual = @fopen(ROOT_DIR . '/.htaccess', 'r' );
4    $nuevo = "";
5    //Searching line to remove
6    while(!feof($actual)) {
7      $linea = fgets($actual);
8      $pos = strpos($linea, "Deny from ");
9      $end = 0;
10     if ($linea === "###WP-Secuirty-Pl###\n" or $linea === "#Blocking IP
       address\n" or $linea === "Order deny,allow\n" or $linea === "\n" and !
       $end) {}
11     else if ($linea === "###End Of WP-Security plugin\n") $end = 1;
12 ...
```

```
...
  else {
      if ($pos != " ") {
        $nuevo .= $linea;
      }
    }
  }
  @fclose($actual);
  $nuevo .= "\n";
  $f = @fopen(ROOT_DIR . '/.htaccess', 'w+' );
  @fwrite($f, $nuevo);
  @fclose($f);
}

function remove_all_related_to_xmlrpc() {
  define (ROOT_DIR, get_home_path() );
  $actual = @fopen(ROOT_DIR . '/.htaccess', 'r' );
  $nuevo = "";
  //Searching line to remove
  while(!feof($actual)) {
    $linea = fgets($actual);
    $posXMLRPC = strpos($linea, "Allow from ");
    $endXMLRPC = 0;
    if ($linea === "# Block WordPress xmlrpc.php requests\n" or $linea ===
    "<Files xmlrpc.php>\n" or $linea === "Order deny,allow\n" or $linea ===
     "Deny from all\n" or $linea === "\n" and !$endXMLRPC) {}
    else if ($linea === "</Files>\n") $endXMLRPC = 1;
    else {
      if ($posXMLRPC != " ") {
        $nuevo .= $linea;
      }
    }
  }
  @fclose($actual);
  $f = @fopen(ROOT_DIR . '/.htaccess', 'w+' );
  @fwrite($f, $nuevo);
  @fclose($f);
}
```

Cuadro 11: Código PHP realizado para eliminar las líneas del fichero *.htaccess*.

3.3. Inicio de sesión

Una vez se han realizado las funciones de activación y desactivación del plugin, las siguientes funcionalidades importantes a realizar son las que están relacionadas con el inicio de sesión del usuario en el panel de administración del sitio web WordPress®.

En este caso, no se va a proceder a programar dicha funcionalidad desde cero, ya que WordPress® dispone de ganchos[8] ofrecidos por su API (interfaz de Programación de la Aplicación) que permiten agregar comportamientos o cambiar el funcionamiento por defecto sin tener que modificar directamente el núcleo.

En la actualidad, WordPress® dispone de dos tipos de *hooks*:

- Acciones: son los *hooks* que ejecuta el núcleo de WordPress® en puntos específicos durante la ejecución, o cuando ocurre un evento concreto.
- Filtros: son los *hooks* que ejecuta el núcleo de WordPress® para modificar textos antes de añadirlos a la base de datos o enviarlos a la pantalla del navegador.

Por último, indicar que los *hooks* en un plugin funcionan de la siguiente manera: mientras WordPress® se está ejecutando, comprueba si algún plugin ha registrado alguna función para ejecutarse en ese preciso instante, y si es así, ejecuta dicha función.

Una vez explicado el funcionamiento de los *hooks*, ya se está en posición de explicar las funcionalidades que se han diseñado relacionadas con el inicio de sesión en el panel de administración.

Empezando por la primera modificación, en este caso, no se cree correcto el funcionamiento que tiene WordPress® por defecto en el panel de acceso (o login), ya que éste muestra cuando el usuario utilizado para iniciar sesión existe o no, es decir, en el caso de que se introduzcan unas credenciales incorrectas, éste devuelve si el usuario no existe, y en el caso de que si exista, muestra que la contraseña introducida es incorrecta. Así pues, se ha modificado el funcionamiento para que cuando se introduzcan unas credenciales incorrectas muestre ambiguedad: `Usuario o Contraseña incorrectos`. Esto se ha hecho creando una función que modifica el funcionamiento habitual mediante el filtro *authenticate* que nos ofrece WordPress®: add_filter('authenticate', 'wp_authenticate_username_password_wpsec' ...);

[8] En la terminología de WordPress® se utiliza generalmente, el termino en ingles *hooks* para denominar a los ganchos.

```
...
  if ( !$usuario ) {
    return new WP_Error('no_username', sprintf(__('<strong>ERROR:</strong>
    Wrong User or Password. <a href="%s" title="Recover password">Recover
    password</a>'), site_url('wp-login.php?action=lostpassword', 'login')))
    ;
  }

  if ( !wp_check_password($password, $usuario->data->user_pass, $usuario->
    ID) ) {
    return new WP_Error('incorrect_password', sprintf(__('<strong>ERROR:</
    strong> Wrong User or Password. <a href="%s" title="Recover password">
    Recover password</a>'), site_url('wp-login.php?action=lostpassword', '
    login')));
  }
...
```

Cuadro 12: Código PHP realizado para la autenticación del usuario y contraseña.

3.3.1. Envío de correo electrónico

Este mismo filtro es el que se utiliza también para realizar el envío de correos electrónicos cuando inicia sesión un usuario administrador. En este caso, se ha de mirar el rol del usuario, y si es administrador, se procede a realizar el envío de correos.

```
...
  if (is_super_admin( $usuario->ID ) || is_admin( $usuario->ID )){
    send_mail($username);
  }
...
```

Cuadro 13: Código PHP realizado para el envío de correo.

En el momento de diseñar la función auxiliar para el envío de correos, se consideraron varias opciones. Por un lado, se estudio la posibilidad de enviar los correos electrónicos de forma externa, es decir, utilizando un servidor de correo externo. Por otro lado, se estudió la posiblidad de enviar los correos electrónicos a través de un servidor de correo instalado en el propio servidor.

El hecho de utilizar un servidor de correo externo, supone que el usuario deba iniciar

sesión en su cuenta de correo para poder proceder, y es por eso que, primeramente, se creó un formulario para que el usuario introdujese dichas credenciales. Pero esto supone un problema, y es que, dependiendo de la cuenta de correo utilizada, la configuración de acceso cambia, y puede variar el puerto por el que se accede, o el tipo de autenticación a utilizar. Así pues, finalmente se decidió instalar un servidor de correo en el propio servidor y enviar los correos desde el servidor de correo local.

Por otro lado, se estudió también la posibilidad de si el correo se enviaba solo a la cuenta de correo que se puso durante la instalación WordPress® o enviarlo a todos los usuarios administradores. Finalmente se decidió esta última opción.

3.3.2. Usuario no autorizado

Uno de los ataques más conocidos, y puede que también de los más sufridos que se pueden experimentar hoy en día, es el intento de inicio de sesión por fuerza bruta. Sin entrar en excesivos detalles, el ataque consiste en intentar iniciar sesión con ciertos usuarios conocidos e introducir contraseñas utilizando un diccionario.

Dado que se puede detectar el tipo de usuario con el que se está iniciando sesión, es evidente que, también se puede obtener el nombre de usuario. De esta manera, se ha creado una tabla en la base de datos para almacenar usuarios conocidos, y que si un atacante remoto inicia sesión con alguno de los usuarios que se encuentra en dicha lista, se procede al bloqueo de su dirección IP. La condición que se debe comprobar para ello es muy sencilla: si el usuario se encuentra en al lista y su dirección IP no se encuentra en la lista de direcciones IP que no se deben bloquear, se procede a su bloqueo:

```
1 ...
2   if ( !$usuario ) {
3     ...
4     if (isBlacklisted($username) and !isInWhiteList ($_SERVER['REMOTE_ADDR'
      ])) {
5       ... // Block IP
6     }
7   }
8 ...
```

Cuadro 14: Código PHP realizado para el envío de correo.

3.3.3. Límite de intentos fallidos superado

Aunque el usuario con el que se esté intentando iniciar sesión no se encuentre en la lista de usuarios a bloquear, otro caso que se debe tener en cuenta es, cuando se esté realizando un ataque de fuerza bruta con un usuario válido. Es por eso que, se debe restringir el número de intentos y proceder al bloqueo de la dirección IP haya sobrepasado dicho límite. Para ello, se continúa utilizando el mismo filtro que se ha ido comentando hasta ahora, dentro de la función de autenticación.

```
1 ...
2   if ( !$usuario ) {
3     if (!isInWhiteList($_SERVER['REMOTE_ADDR'])) wp_sec_bloquea_ip_auto();
4     ...
5   }
6 ...
```

Cuadro 15: Código PHP realizado para el bloqueo automático de direcciones IP.

Tal y como se visualiza en el código mostrado 15, se hace uso de la función auxiliar *wp_sec_bloquea_ip_auto()*, que se encarga de almacenar la dirección IP que intenta iniciar sesión y su número de intentos. Cuando la dirección IP llega al límite de intentos fallidos, ésta se encarga de llamar a otra función auxiliar que proceda a su bloqueo.

3.4. Control de peticiones POST

Otro aspecto importante a tener en cuenta, son las peticiones POST que recibe el servidor web, ya que muchos atacantes remotos intentan subir ficheros con código malicioso. Basándose en esta premisa, se ha diseñado una funcioń que monitoriza este tipo de peticiones, y cuando se registra una petición POST a un recurso inexistente, se bloquea la dirección IP en el fichero *.htaccess*, inhabilitando el acceso al sitio web para evitar futuros incidentes de seguridad. El código para capturar este tipo de peticiones es sencillo, pero necesitamos de un nuevo *hook* para poder interactuar con la instalación WordPress®:

```
add_action('wp', 'wp_sec_bloquea_ip_post_404');

function wp_sec_bloquea_ip_post_404() {
  if (is_404() and $_SERVER['REQUEST_METHOD'] === 'POST') {
    ... //Block IP
  }
}
```

Cuadro 16: Código PHP realizado para el bloqueo de direcciones IP que hacen peticiones POST.

3.5. Interfaz

Dado que este plugin ofrece la posibilidad de bloquear o desbloquear direcciones IP manualmente o, añadir una dirección IP a una lista blanca para que no sea bloqueada, o permitir el acceso a una dirección IP mediante XML-RPC, u obtener información sobre una dirección IP (*whois*), entre otras, es indudable pensar que, no disponga de una interfaz con la que el usuario pueda interactuar de manera directa. Igual que en los casos anteriores, WordPress® dispone de un *hook* que permite añadir entradas en el menú de administración.

En este caso, se ha generado una nueva entrada de menú, llamada *WP Security Plugin*. Se muestra el código creado para añadir dicha entrada en el panel de administración:

```
function wp_sec_add_menu(){
  if (function_exists('add_options_page')) add_options_page('WP Security
    Plugin', 'WP Security Plugin', 8, basename(__FILE__), 'wp_sec_pl_panel'
    );
}
if (function_exists('add_action')) {
  add_action('admin_menu', 'wp_sec_add_menu');
}
```

Cuadro 17: Código PHP realizado para agragar una entrada de menú.

Una vez se dispone de la entrada de menú, ya es posible diseñar y realizar la interfaz de usuario, la cual, se encuentra diseñada en lenguaje de marcas de hipertexto (HTML). Se debe aclarar que, dado que no es el objetivo principal del proyecto describir en la memoria el proceso seguido para el desarrollo de la interfaz, en esta sección se describirán

simplemente un par de cuestiones importantes. Por otro lado, se dispone del anexo (Añadir referencia), en el cual se muestran capturas de pantalla de la interfaz.

Una de las cuestiones que se debe clarificar es cómo el usuario se comunica con el plugin, es decir, qué método utiliza para facilitar la información al plugin para que éste la gestione. En éstos casos, el usuario proporciona la información al plugin a través de formularios, que por un lado, se almacena en tablas de la base de datos, además de derivar en las acciones pertinentes, ya sea, bloquear una dirección IP o mostrar determinada información.

Por otro lado, el plugin debe mostrarle al usuario toda la información recopilada hasta el momento, ya que éste actúa de forma autónoma y, es comprensible que, el usuario desee conocer si se ha realizado algún cambio en la base de datos, es decir, si se ha bloqueado alguna dirección IP, si se han permitido acceso vía XML-RPC, etc...

4 Pruebas en un entorno real

Tras haber finalizado el desarrollo del plugin, el siguiente paso que se debe realizar es, instalarlo y ejecutarlo en un entorno real para comprobar su correcto funcionamiento. Por lo tanto, se ha realizado la instalación WordPress® en un servidor virtual privado (VPS) – proviene del inglés: (*Virtual Private Server*) – en el que se ha instalado Apache como servidor web, MySQL como base de datos y Postfix como servidor de correo.

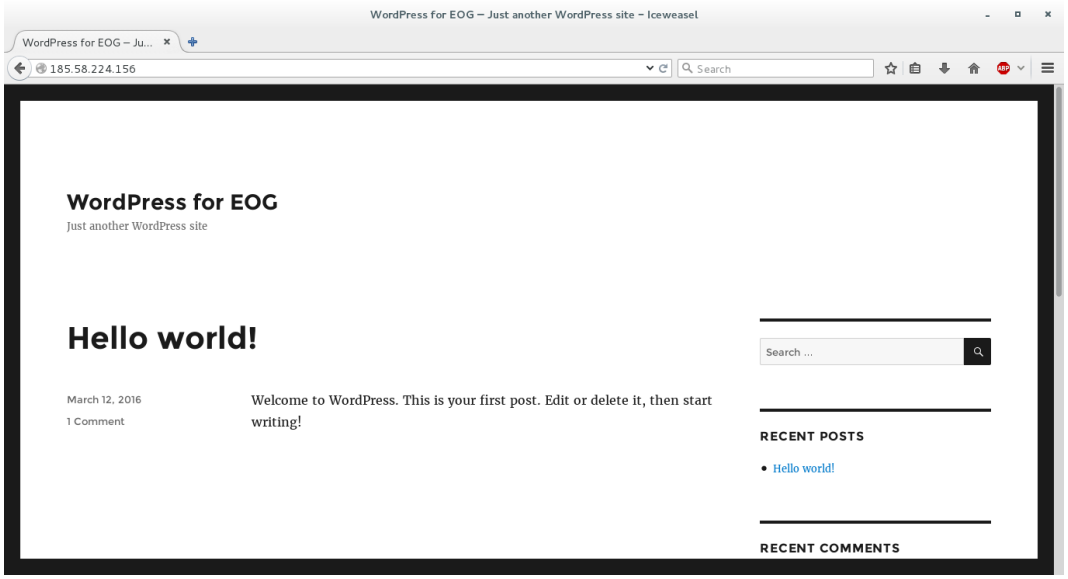

Figura 6: Captura de pantalla que muestra WordPress® funcionando en el VPS.

Una vez tenemos ya la instalación realizada con éxito y deseamos realizar modificaciones en las diferentes opciones, iniciamos sesión en el panel de administración. Paralélamente, accedemos a la cuenta de correo que se indicó durante la instalación, y observamos que se ha recibido un nuevo correo en la bandeja de entrada:

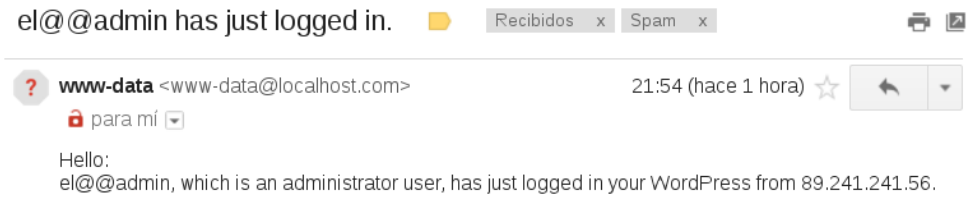

Figura 7: Captura de pantalla que muestra el correo electrónico recibido.

Por otro lado, disponemos de una lista de usuarios que se han añadido a la lista negra o *Blacklist*, y que por tanto, provocarán el bloqueo de aquella dirección IP que intente iniciar sesión con uno de ellos:

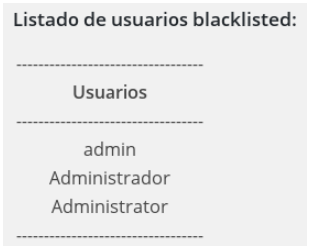

Figura 8: Captura de pantalla que muestra la lista de usuarios en lista negra.

A continuación, se muestra la lista blanca o *Whitelist*, en la que se encuentran todas aquellas direcciones IP que no procederán a ser bloqueadas bajo ningún concepto:

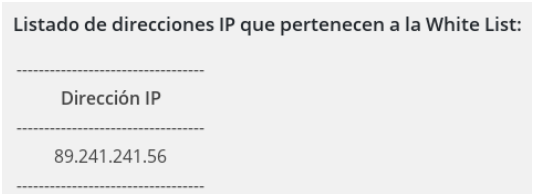

Figura 9: Captura de pantalla que muestra la lista de direcciones IP incluidas en la Lista blanca.

Además, también podemos encontrar una lista con direcciones IP que han sido incluidas en la lista negra, ya sea de forma manual o automática:

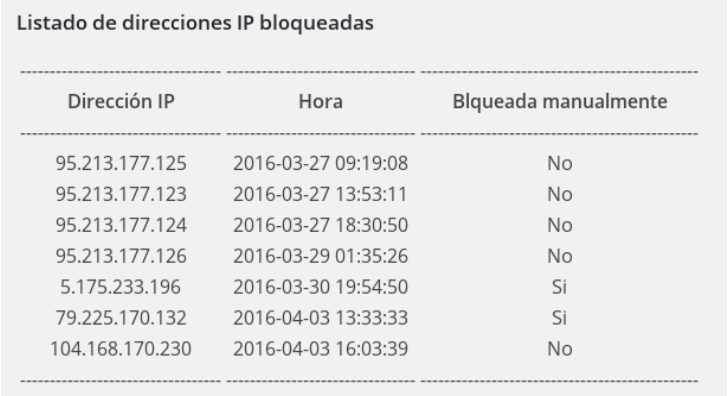

Figura 10: Captura de pantalla que muestra la lista de direcciones IP incluidas en la Lista blanca.

En relación con la última captura mostrada 10, si visualizamos los logs del servidor web Apache, veremos que efectivamente, la respuesta que ha registrado el servidor para la dirección IP 95.213.177.124, ha cambiado entre las diferentes peticiones que ha realizado:

```
root@VPSUK-WP:/var/log/apache2# cat access.log.7 | grep -v "89.241.241.56" | grep -
v "104.168.170.230" | grep -v "::1 - -" | grep -v "185.58.224.156" | grep "95.213.1
77.124"
95.213.177.124 - - [27/Mar/2016:20:30:50 +0200] "POST http://check.proxyradar.com/a
zenv.php?auth=145910345029&a=PSCMN&i=3107643548&p=80 HTTP/1.1" 404 9818 "https://pr
oxyradar.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"
95.213.177.124 - - [28/Mar/2016:02:27:52 +0200] "POST http://check.proxyradar.com/a
zenv.php?auth=145912487267&a=PSCMN&i=3107643548&p=80 HTTP/1.1" 403 382 "https://pro
xyradar.com/" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"
root@VPSUK-WP:/var/log/apache2#
```

Figura 11: Captura de pantalla que muestra las respuestas del servidor web.

Otro ejemplo para mostrar que se le ha denegado el acceso a una dirección IP, en concreto la dirección IP 104.168.170.230, es que cuando se intenta acceder, visualiza esta respuesta en su navegador web:

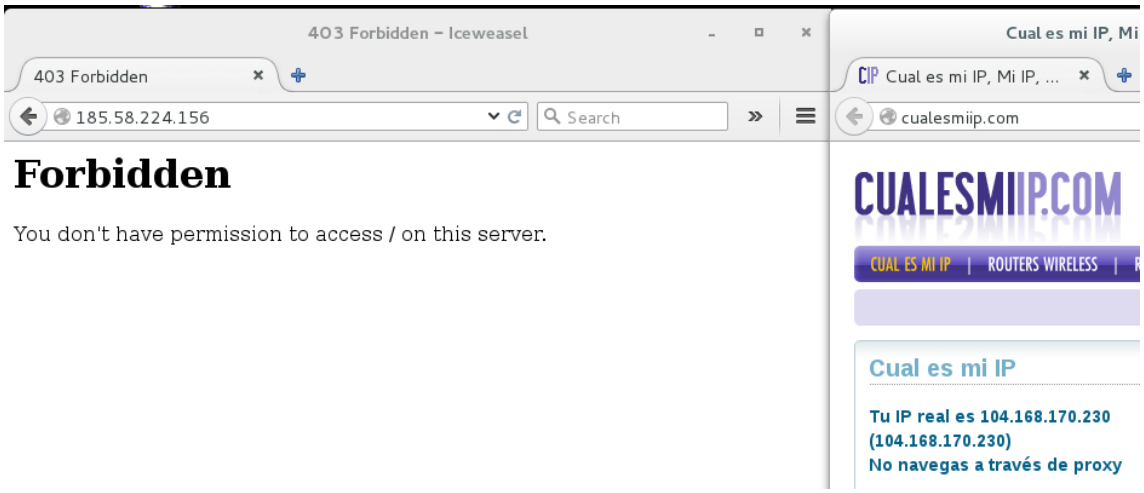

Figura 12: Captura de pantalla que muestra el navegador web del usuario.

Seguidamente, se muestra una lista con direcciones IP que tienen acceso a la instalación WordPress® mediante XML-RPC:

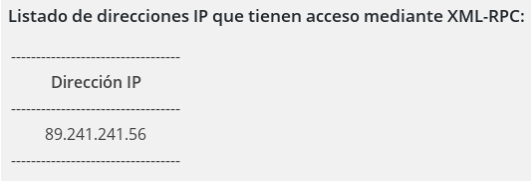

Figura 13: Captura de pantalla que muestra la dirección con acceso mediante XML-RPC.

Como se puede visualizar en las siguientes capturas, la dirección IP 89.241.241.56 sí tiene acceso, mientras que la dirección IP 104.168.170.230, así como el resto de direcciones, no lo tienen:

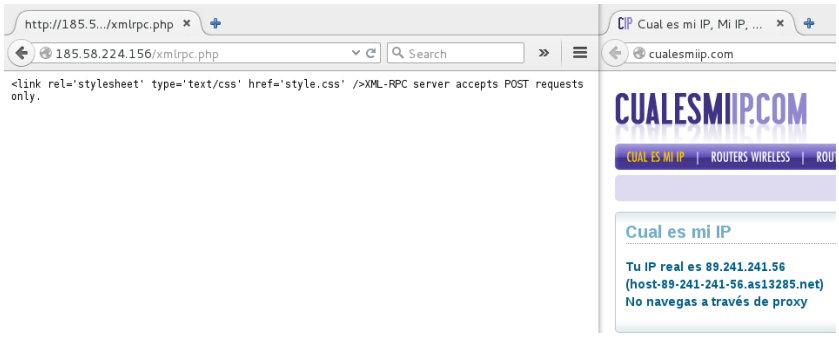

Figura 14: Captura de pantalla que muestra las direcciones con acceso mediante XML-RPC.

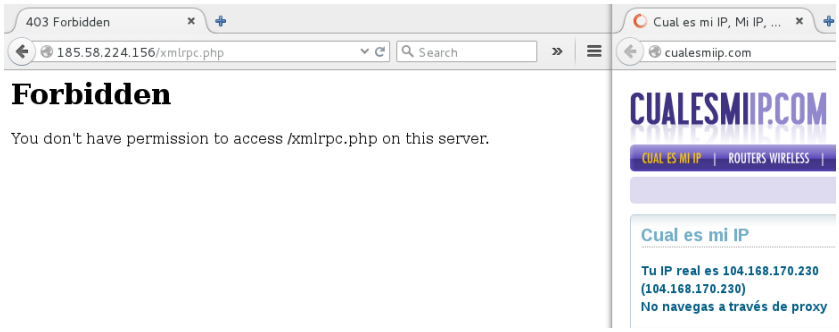

Figura 15: Captura de pantalla que muestra las direcciones con acceso mediante XML-RPC.

Por último, en la siguiente captura de pantalla, se muestra la herramienta de *Whois* en funcionamiento:

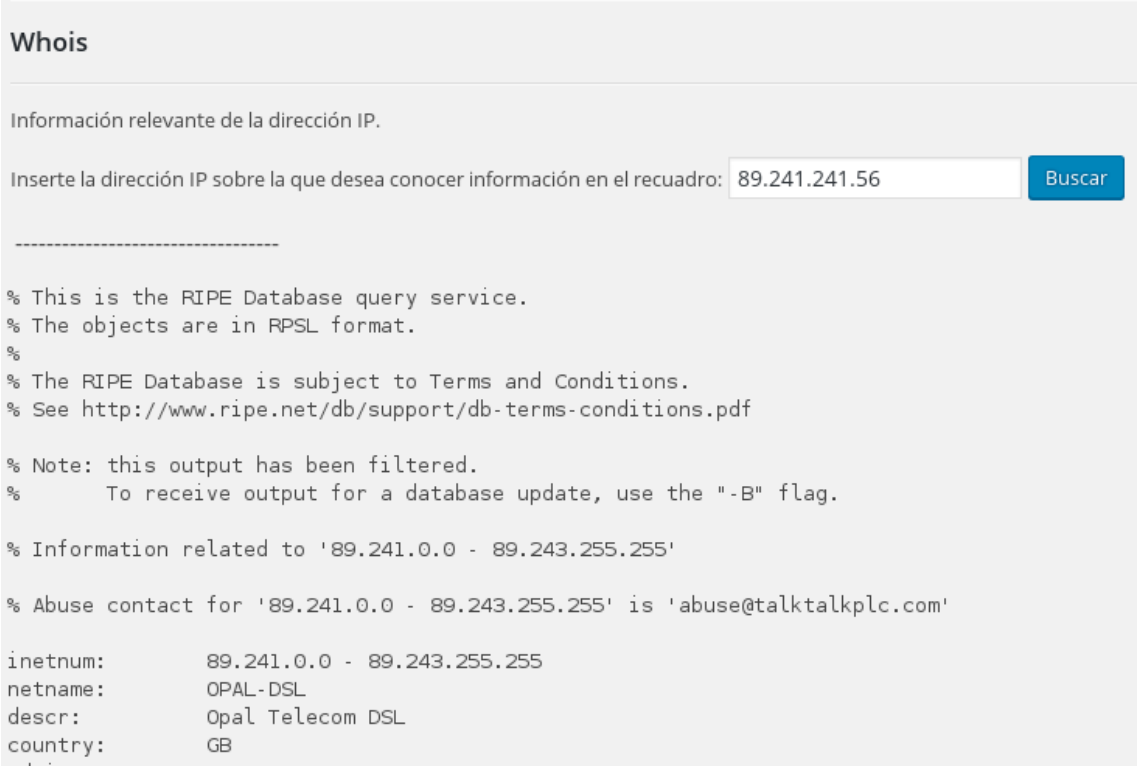

Figura 16: Captura de pantalla que muestra la herramienta de *Whois*.

5 Conclusiones

WordPress® es un gestor de contenidos al que se le da un gran uso actualmente. Debido a esto, en mi trayectoria profesional, he tenido la oportunidad de profundizar en la securización de los mismos, y es por eso que, dicho CMS ha generado en mí un gran interés.

Con este proyecto he estudiado el funcionamiento interno de WordPress® y he creado un plugin que proporciona seguridad al mismo, evitando que se puedan producir situaciones que puedan suponer un riesgo y un futuro incidente de seguridad.

El desarrollo del plugin ha supuesto un gran reto para mí, ya que nunca antes había tenido que enfrentarme a ningún caso similar. He tenido que invertir mucho tiempo en documentarme sobre el gestor de contenidos, en conocer las buenas prácticas para desarrollar un plugin y en conocer y dominar algunas funciones que proporciona la API de WordPress®, sobretodo el uso de *hooks*. Incluso, ha sido necesario profundizar tanto en la sintáxis del propio lenguaje PHP, como en la del lenguaje de marcas de hipertexto (HTML), o en el uso de sentencias SQL. También se ha dedicado tiempo a investigar cómo se realiza una instalación segura del entorno donde se va a ejecutar WordPress®, para evitar así, que puedan explotarse vulnerablidades conocidas, por ejemplo, del servidor web.

Esto me ha hecho ver, que el querer profundizar en un aspecto en el mundo de la informática, posiblemente, implique profundizar también en el resto de herramientas que se necesitan para su ejecución. Creo que es un punto importante a tener en cuenta, sobretodo cuando se estudia el alcance del proyecto.

Por otro lado, el desarrollo de un plugin es un tema muy amplio, igual que el tema de la seguridad. Esto implica que, este proyecto esté abierto a investigaciones futuras y mejoras, sobretodo en el aspecto de la funcionalidad.

Para concluir este apartado, añadir que el esfuerzo y el tiempo que se ha dedicado al desarrollo del proyecto, han merecido la pena. Ha habido situaciones difíciles que se han tenido que ser superadas para aportar buenas soluciones a los objetivos propuestos, pero en general la sensación definitiva del proyecto es muy positiva.

5.1. Trabajos futuros

En este apartado se exponen una serie de funcionalidades que por su extensión excedían los objetivos del proyecto, y que considero de gran interés, por lo que se proponen como futuros proyectos.

- Poder modificar el número de intentos fallidos a los que se desea bloquear una dirección IP, dado que la solución provista en este proyecto solo permite bloquear una dirección IP a los tres intentos de inicio de sesión fallidos.
- Aplicar el bloqueo a una dirección IP de manera temporal, pudiendo así, escoger el tiempo que permanecerá sin acceso al sitio web.
- Informarse de cómo funciona el sistema de actualizaciones de los plugins en la tienda de WordPress®, para que los usuarios que deseen instalarse y disfrutar de las ventajas que ofrece dicho plugin, dispongan siempre de la última versión y sus últimas novedades.
- Realizar los archivos necesarios (.po y .mo) para conseguir que el plugin sea multilenguaje, y así conseguir un mayor público.
- Hacer compatible el plugin para servidores nginx, ya que es posible ejecutar WordPress® también en este tipo de servidores.

A Instalación de WordPress®

En este primer anexo se procede a explicar cómo realizar la instalación de WordPress® en un entorno GNU Linux con sistema operativo debian en su versión 8.0 (*Jessie*).

A.1. Instalación del servidor web

Para instalar el servidor web Apache en debian, simplemente es necesario ejecutar por terminal el siguiente comando:

```
1 root@U30Jc:~# aptitude install apache2
```

Cuadro 18: Comamando a ejecutar en la terminal para instalar el servidor web Apache.

A.2. Instalación de la base de datos MySQL

Seguidamente, para la instalación de la base de datos MySQL, se instala mediante el sigguiente comando en terminal:

```
1 root@U30Jc:~# aptitude install mysql-server
```

Cuadro 19: Comando a ejecutar en la terminal para instalar la base de datos MySQL.

A.3. Instalación del intérprete PHP

Por otro lado, tal y como se ha explicado durante el desarrollo del proyecto, el núcleo del gestor de contenidos está diseñado en lenguaje PHP. Es por eso que, se necesita un intérprete en el activo servidor. Los paquetes que debemos instalar, en este caso, mediante la terminal son los siguientes:

```
1 root@U30Jc:~# aptitude install php5 php5-mysql
```

Cuadro 20: Comando a ejecutar en la terminal para instalar el intérprete de PHP.

A.4. Instalando WordPress®

Ahora que ya tenemos descargado todo el entorno necesario para ejecutar el gestor de contenidos, debemos proceder a la descarga e instalación del mismo.

Tal y como se explica en [6], lo primero que debemos hacer, es proceder a descargarlo desde su página web oficial. Igual que en el resto de comandos explicados anteriormente, se puede realizar directamente desde la terminal, pero también es posible descargarlo mediante el navegador web. Si lo hacemos directamente desde el navegador, simplemente debemos incluir este enlace: http://wordpress.org/latest.tar.gz en la barra de navegación y se procederá a su descarga de manera automática. Por otro lado, para realizar la descarga desde la terminal, debemos ejecutar el siguiente comando:

```
1 root@U30Jc:~# wget http://wordpress.org/latest.tar.gz
```

Cuadro 21: Comando a ejecutar en la terminal para descargar la última versión de WordPress®.

Una vez lo tenemos descargado, procedemos a cambiar el nombre del archivo para trabajar con mayor comodidad y lo descomprimimos. Podemos realizar estas acción mediante la interfaz o por terminal. Los comandos para realizar dichas acciones utilizando la terminal son:

```
1 root@U30Jc:~# mv wordpress-4.3.1.tar.gz wordpress.tar.gz
2 root@U30Jc:~# tar -xzvf wordpress.tar.gz
```

Cuadro 22: Comandos para cambiar el nombre y descomprimir el archivo descargado desde la terminal.

Una vez ya obtenemos la carpeta, se mueve al directorio correspondiente y se le dan los permisos adecuados para que se ejecute correctamente en el servidor. En este caso, dado que se va a ejecutar la instalación en un servidor debian, la ruta es: */var/www/html/*. El siguiente paso es dar los permisos apropiados para que el servidor web pueda leer y escribir sobre los archivos que se acaban de mover en el directorio, y para ello se debe escribir en la terminal el siguiente comando:

```
root@U30Jc:/var/www/html# chown www-data:www-data /var/www/* -R
```

Cuadro 23: Muestra de cómo aplicar permisos apropiados a la instalación WordPress®.

En este caso, se ha decido que el usuario *www-data* tenga permisos de escritura sobre el directorio para que no sea necesario usar credenciales de acceso por FTP o SFTP cuando se desee realizar cualquier instalación o actualización en el gestor de contenidos WordPress®.

Una vez ya tenemos el WordPress® instalado y colocado en la ruta deseada y con los permisos adecuados, debemos proceder a la creación de la base de datos donde se almacenarán todos los contenidos generados en el WordPress® instalado. La creación de la base de datos se realiza mediante el software instalado de mysql de la siguiente manera:

```
root@U30Jc:/var/www/html# mysql -u root -h localhost -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42
Server version: 5.5.44-0+deb8u1 (Debian)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>
mysql> CREATE DATABASE Wordpress;
mysql> CREATE USER edgar@localhost;
mysql> SET PASSWORD FOR edgar@localhost= PASSWORD("XXXXXX");
mysql> GRANT ALL PRIVILEGES ON Wordpress.* TO edgar@localhost IDENTIFIED BY 'XXXXXX
    ';
mysql> FLUSH PRIVILEGES;
mysql> quit
```

Cuadro 24: Configuración de la base de datos MySQL para WordPress®.

Por último, una vez tenemos diseñada ya la base de datos hemos de modificar las líneas correspondientes del archivo *wp-config.php* 25 de acuerdo a las acciones realizadas.

```
1  // ** Ajustes de MySQL. Solicita estos datos a tu proveedor de alojamiento
       web. ** //
2  /** El nombre de tu base de datos de WordPress */
3  define('DB_NAME', 'Wordpress');
4
5  /** Tu nombre de usuario de MySQL */
6  define('DB_USER', 'edgar');
7
8  /** Tu contrasena de MySQL */
9  define('DB_PASSWORD', 'XXXXXX');
10
11 /** Host de MySQL (es muy probable que no necesites cambiarlo) */
12 define('DB_HOST', 'localhost');
```

Cuadro 25: Muestra de la configuración relevante del fichero *wp-config.php*.

B Descripción de la interfaz

Este segundo anexo aporta una descripción mediante capturas de pantalla, de lo que un usuario observará cuando acceda al plugin desde el panel de administración del plugin.

Lo primero que el usuario visualiza cuando accede, es una pequeña descripción de las funciones con las cuales, podrá interactuar con el plugin para transmitirle la información.

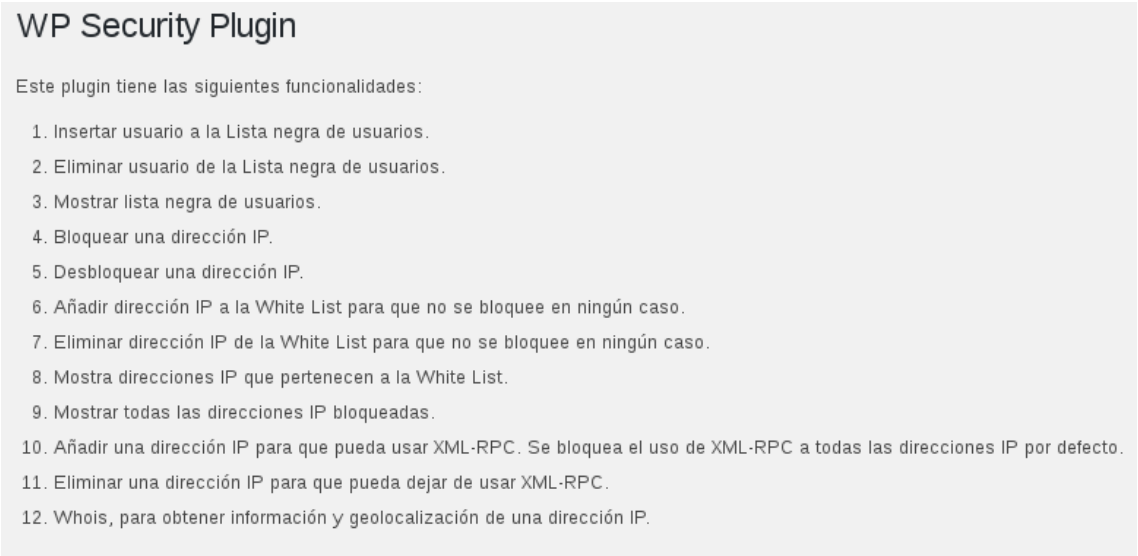

Figura 17: Captura de pantalla que muestra una descripción de las funcionalidades.

Justamente después de esta descripción, existe otra descripción de las funcionalidades que el plugin ejecuta de manera autónoma, sin que el usuario necesite interactuar con el mismo.

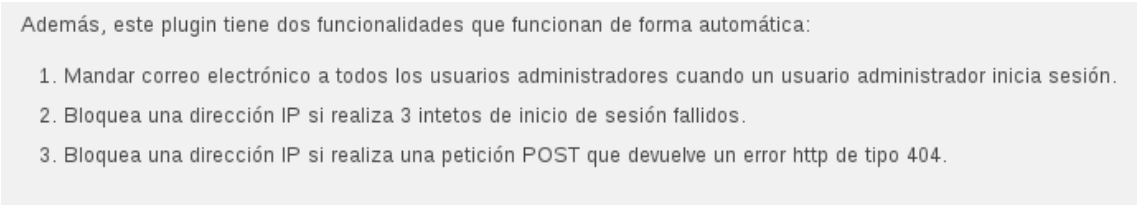

Figura 18: Captura de pantalla que muestra una descripción de las funcionalidades.

Una vez, ya han finalizado todas las descripciones de las funcionalidades que el plugin puede realizar, se muestran las acciones con las que el usuario puede interactuar.

Las primeras gestiones que el usuario puede realizar, son referentes al manejo de usuarios no permitidos. El usuario podrá añadir usuarios, eliminar y visualizar una lista de los usuarios que se encuentran includios en dicha lista:

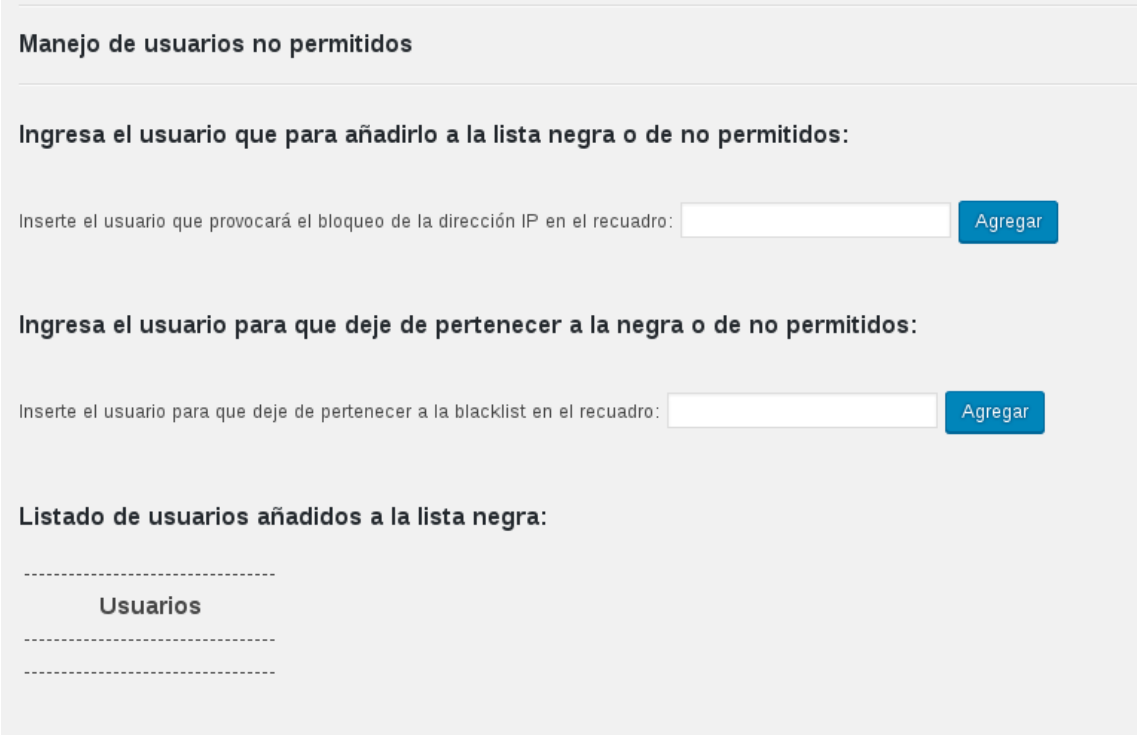

Figura 19: Captura de pantalla que muestra el manejo de usuarios.

Seguidamente, el usuario podrá visualizar las opciones para realizar las gestiones de direcciones IP, es decir, añadir una dirección IP a la lista negra para bloquearla, eliminarla de dicha lista. También podrá añadir una dirección IP a la lista blanca, para que no se proceda a su bloqueo bajo ningún concepto. De la misma manera, podrá eliminar una direccioń IP de dicha lista. Se muestra dicha captura en la figura 20.

Además de añadir o eliminar direcciones IP de una lista o de otra, el usuario podrá visualizar las direcciones IP que están incluidas en alguna de ambas listas. Se muestra dicha captura en la figura 21.

Figura 20: Captura de pantalla que muestra el manejo de direcciones IP.

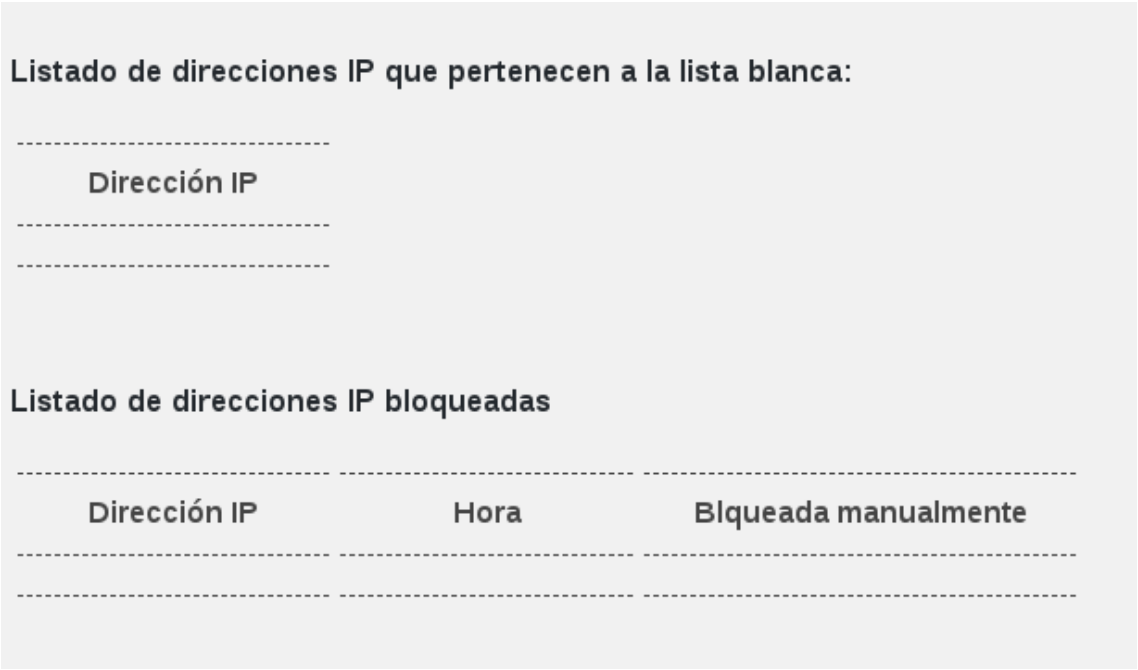

Figura 21: Captura de pantalla que muestra los listados referentes a las listas de direcciones IP.

Continuando con las opciones que ofrece este plugin, igual que en el caso del manejo de usuarios y de las direcciones IP, el usuario podrá gestionar el acceso de las direcciones IP mediante XML-RPC. Dado que por defecto, no se permite el acceso mediante XML-RPC a ninguna dirección IP, el usuario deberá añadir una a una las direcciones que desea permitir. De la misma manera, puede eliminar una dirección IP para que deje de tener acceso. Posteriormente, se le mestra la lista con las direcciones IP que tienen dicho acceso permitido:

Figura 22: Captura de pantalla que muestra el manejo de acceso mediante XML-RPC.

Por último, el usuario dispone de la herramienta de *Whois*, con la que podrá obtener información de una dirección IP directamente desde la interfaz del plugin, sin necesidad de utilizar herramientas de terceros:

Figura 23: Captura de pantalla que muestra la herramienta de *Whois*.

C Bloqueo de direcciones IP mediante un script

El fin con el que se realiza este script, es para proporcionar una mayor seguridad al servidor, bloqueando una dirección IP de forma permanente en el Firewall, ya que el hecho de bloquearla mediante una directiva en el fichero *.htaccess* no asegura que un atacante continúe realizando peticiones maliciosas al resto de dominios alojados en éste o intentando explotar algún otro tipo de vulnerabilidad.

Para ello, a la hora de realizar el script se han tenido en cuenta las peticiones POST que puede realizar un atacante al recurso para acceder al panel de administración de WordPress® (*/wp-login.php*) y, que devuelven una respuesta http de error, de tipo 403. Teniendo en cuenta este tipo de peticiones, evitamos bloquear de forma permanente a un posible usuario que se hubiera bloqueado accidentalmente en el fichero *.htaccess*, ya que éste realizaría peticiones GET, aunque también recibiera una respuesta de error, de tipo 403.

Basándose en esta premisa, se realiza una búsqueda de este tipo de peticiones en los logs del servidor web. Una vez se ha obtenido una lista de direcciones IP, se comprueba que no se encuentran bloqueadas ya en el Firewall, y si no es así, se procede a bloquearlas.

Por último, se debe añadir que en este caso, se ha diseñado para un Firewall específico, como es CSF, pero no es difícil adaptarlo para otros.

C.1. Código realizado

A continuación se muestra el código del script realizado:

```
1 egrep '"POST /wp-login\.php HTTP\/1\.0" 403' /var/log/apache2/*/access.log* | grep
    -v ftp_log | awk -F : '{print $1" "$2}' | awk '{print $1" "$2}' | sort | uniq -
    c | (
2 while read a b f
3 do
4   if test $a -gt 0
5   then
6     if ! /bin/grep $b /etc/csf/csf.deny  /etc/csf/csf.ignore >/dev/null 2>/dev/
    null
7     then
8       geo=$(/usr/bin/geoiplookup $b |  /bin/awk -F: '{print $2}' | /usr/bin/paste
    -sd "," - )
9 ...
```

```
...
    echo Filtrada la direccion IP $b pertenciente a $geo con $a accesos a wp-login
       /usr/sbin/csf -d $b $geo wordpress blocked wp-login
     fi
  fi
done
)
```

Cuadro 26: Muestra del código del script realizado.

C.2. Pruebas de funcionamiento

Una vez se ha acabado de diseñar el código, y para comprobar que, el código funciona correctamente, se ejecuta el script manualmente en el servidor VPS con el que se han realizado las pruebas del plugin y en el que se encuentra la instalación WordPress®. Éstos son los resultados obtenidos:

```
root@VPSUK-WP:/home/emontanes# ./bloqueaIP.sh
Filtrada la direccion IP 31.48.8.113 que pertenece a UK con 1 accesos a wp-login
csf -d 31.48.8.113 UK wordpress blocked wp-login
Filtrada la direccion IP 5.175.233.196 que pertenece a DE con 2 accesos a wp-login
csf -d 5.175.233.196 DE wordpress blocked wp-login
root@VPSUK-WP:/home/emontanes#
```

Cuadro 27: Muestra de la respuesta tras ejecutar el script.

Referencias

[1] WordPress®:

https://es.wikipedia.org/wiki/WordPress

Acceso: 29 de septiembre de 2015.

[2] Sistema de gestión de contenidos:

https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_contenidos

Acceso: 29 de septiembre de 2015.

[3] Top WordPress security plugins 2015:

https://www.linkedin.com/pulse/top-wordpress-security-plugins-2015-your-website-stanley

Acceso: 29 de septiembre de 2015.

[4] Wordfence Security:

https://wordpress.org/plugins/wordfence/

Acceso: 7 de octubre de 2015.

[5] Wordfence Security:

https://www.wordfence.com/wordfence-signup/

Acceso: 7 de octubre de 2015.

[6] Instalando WordPress®en tu equipo Debian:

https://www.digitalocean.com/community/tutorials/

how-to-install-wordpress-on-debian-7

Acceso: 11 de octubre de 2015.

[7] Bulletproof-security features:

https://wordpress.org/plugins/bulletproof-security/

Acceso: 25 de octubre de 2015.

[8] Login LockDown:

https://wordpress.org/plugins/login-lockdown/

Acceso: 25 de octubre de 2015.

[9] iThemes Security (formerly Better WP Security):

https://es.wordpress.org/plugins/better-wp-security/

Acceso: 25 de octubre de 2015.

[10] Brad Williams, Ozh Richard, and Justin Tadlock.
Professional WordPress Plugin Development.
Wiley Publishing, Inc., Indianapolis, 2011. ISBN: 978-0-470-91622-3.

[11] How To Install WordPress on Debian 7 — DigitalOcean:
https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-on-debian-7
Acceso: 29 de septiembre de 2015.

[12] Escribiendo un plugin:
https://codex.wordpress.org/Escribiendo_un_Plugin
Acceso: 29 de septiembre de 2015.

[13] How To Create A WordPress Plugin:
https://www.elegantthemes.com/blog/tips-tricks/how-to-create-a-wordpress-plugin
Acceso: 29 de septiembre de 2015.

[14] Cómo crear un plugin para WordPress:
http://www.cristalab.com/tutoriales/como-crear-un-plugin-para-wordpress-c54308l/
Acceso: 29 de septiembre de 2015.

[15] API de WordPress®:
https://codex.wordpress.org/es:Plugin_API
Acceso: 29 de septiembre de 2015.

[16] ¿Qué son y para qué te sirven los hooks de WordPress?:
https://platzi.com/blog/hooks-wordpress/
Acceso: 29 de septiembre de 2015.

[17] Constantes de WordPress – Lista completa y descripción | Ayuda WordPress:
http://ayudawp.com/constantes-de-wordpress/
Acceso: 24 de octubre de 2015.

[18] PHP: shell_exec - Manual:
http://php.net/manual/es/function.shell-exec.php
Acceso: 14 de noviembre de 2015.

[19] Gnoma 10 cosas que quizás no sabias de PHP en la shell:
http://gnoma.es/blog/10-cosas-que-quizas-no-sabias-de-php-en-la-shell/
Acceso: 14 de noviembre de 2015.

[20] Cómo hacer un abstract:
http://es.slideshare.net/solartime/como-hacer-un-abstract
Acceso: 14 de noviembre de 2015.

[21] Importante para hacer un buen abstract:
http://www.ipnm.edu.pe/web/images/stories/publicaciones/Abstract.pdf
Acceso: 14 de noviembre de 2015.

[22] Bastionado de un servidor web Apache:
http://www.securityartwork.es/2012/02/22/bastionado-de-un-servidor-web-apache-i/
Acceso: 16 de enero de 2016.

[23] 20 trucos para configurar *.htaccess*:
http://ignaciosantiago.com/blog/web/seguridad-en-wordpress-como-configurar-htaccess-par
Acceso: 16 de enero de 2016.

[24] How to send emails in WordPress:
http://bordoni.me/send-wordpress-email/
Acceso: 6 de febrero de 2016.

[25] Usurios en WordPress: Funciones y Plugins útiles:
http://www.emenia.es/funciones-plugins-usuarios-wordpress/
Acceso: 6 de febrero de 2016.

[26] GNU General Public License:
https://es.wikipedia.org/wiki/GNU_General_Public_License
Acceso: 26 de marzo de 2016.