

WHITEPAPER_

Wordpress in Paranoid Mode (Parte 1)

Chema Alonso (chema@11paths.com)

Pablo González (pablo@11paths.com)

02.11.2016

Índex

1. Amenazas contra la identidad, confidencialidad, integridad y disponibilidad	4
2. Fortificación a nivel de usuario con 2 Factor Authorization	4
2.1. Instalación del Plugin de Latch para Wordpress.....	5
2.2. Pairing del Plugin de Latch para Wordpress.....	6
2.3. Verificación ante intento de suplantación de identidad en Wordpress	7
3. Fortificación a nivel de tabla con 2 Factor Authorization.....	7
3.1. Modo de actuación	7
3.2. Tablas afectadas	8
3.3. Esquema global de WPM	10
3.4. Algoritmo de los triggers	11
3.5. Proceso de instalación.....	13
3.6. Protegiendo Wordpress ante acciones maliciosas.....	14
Acerca de ElevenPaths	17
Más información	17

Abstract

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 identidad, la disponibilidad e integridad de los activos de una organización/usuario que utilice Wordpress. La protección de la identidad mediante un segundo factor de autorización es fundamental. Por otro lado, la protección de la disponibilidad e integridad de los datos que son almacenados en la base de datos de Wordpress es imprescindible. Ambas medidas multiplican el nivel de seguridad que se proporciona al CMS.



1. Amenazas contra la identidad, 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 que deben ser cubiertas por cualquier mecanismo de protección IT, en cualquier tipo de entorno. Una adecuada gestión de la identidad, mediante la proposición de mecanismos de autenticación robustos y usables, es un pilar fundamental para garantizar estas dimensiones de la seguridad los datos. La autenticación/autorización de un usuario permite a Wordpress determinar si se trata de un usuario legítimo y permite determinar a qué recursos tiene acceso. Cualquier amenaza definida sobre el sistema de gestión identidad es crítico para el estado de seguridad de cualquier sistema IT.

Las amenazas se encuentran, generalmente, en Internet y pueden materializarse en forma de distintos tipos de ataques. Los más comunes persiguen comprometer los mecanismos de autenticación y esto se puede conseguir de diferentes modos:

- Robo de credenciales en ataque a las comunicaciones, por ejemplo, con canales no seguros.
- Robo de credenciales tras la explotación en uno de los plugins instalados en Wordpress, o vulnerabilidades en el propio CMS.
- Robo de credenciales tras suplantación o engaño a través de un ataque de phishing o spear phishing.
- Robo de credenciales tras ataque a equipo cliente troyanizado.

El compromiso de las credenciales asociadas a la identidad de un usuario no solo supone que alguien no autorizado tiene acceso a recursos para los que no tiene autorización. Potencialmente, una derivada de esta brecha de seguridad es que información crítica del usuario o de la organización donde desarrolla su profesión puede verse expuesta en ámbitos desconocidos por la víctima.

La proliferación de servicios como *"haveibeenpwned"* o *"hesidohackeado"* pueden ayudar en una constante vigilancia digital para conocer en qué momento y en qué brecha de seguridad la identidad está expuesta. Lógicamente, estos servicios solo proporcionan información sobre ataques o incidentes importantes, por lo que sí es un ataque aislado o dirigido contra una persona, ésta no podría darse cuenta de que su identidad está expuesta.

2. Fortificación a nivel de usuario con 2 Factor Authorization

Los mecanismos de autenticación deben garantizar que el usuario que solicita el acceso a un recurso es un usuario legítimo. Estos mecanismos se diseñan de tal manera que permiten validar diferentes factores de autenticación de una forma segura. Estos factores de autenticación permiten al usuario demostrar que es quien dice ser. Y para ello existen diferentes estrategias para proponer factores de autenticación. Se pueden considerar factores de autenticación basados en:

- Algo que el usuario conoce. En otras palabras, una contraseña de acceso y que verifique que el usuario la conoce.
- Algo que el usuario tiene. La posesión de un dispositivo móvil o un token RSA permite al sistema que debe verificar la identidad de un usuario disponer de un canal alternativo por dónde comunicarse con él. Uno de los usos más comunes es la generación de tokens o envío de mensajes de texto con un código a través de dichos canales, con el objetivo de que el usuario pueda introducirlo en un segundo paso de autenticación.
- Algo que el usuario es. La biometría ha irrumpido en el mundo de la seguridad y permite que un sistema verifique algún rasgo biométrico de un usuario. De nuevo es paso más en la autenticación.

- Algo que el usuario hace. Al final todos los usuarios interactúan de una forma pautada con los sistemas de información. Esta pauta permite definir un comportamiento esperado que se puede utilizar como factor de autenticación.

Tradicionalmente se consideró que unos factores eran mejores que otros y se forzó la creación de passwords cada vez más complejas o sistemas biométricos cada vez más invasivos. Sin embargo, en los últimos años se ha podido comprobar que es más efectivo reducir la probabilidad de que un ataque tenga éxito usando más de un factor de autenticación antes que usar uno solo muy robusto. Típicamente, el uso de estos factores extra de autenticación implica la definición de un canal extra a través del cual se definen protocolos que garantizan que su validación aumente la fiabilidad del proceso de autenticación. Existen múltiples alternativas para construir estos canales pero, últimamente, el extendido uso de los teléfonos móviles han forzado a que los canales que explotan su uso sean los más usados (SMS, SSL, sistemas de notificaciones, etc.).

Latch es una de estas soluciones que permiten construir un nivel extra de protección a cualquier mecanismo de autenticación/autorización. En realidad, Latch permite incorporar al proceso de validación de credenciales las restricciones que el usuario (user constrains) legítimo ha definido para el uso de un sistema o servicio concreto. A diferencia de otras medidas de protección globales, Latch permite que sea el usuario el que decida cuándo y bajo qué circunstancias el acceso a determinadas operaciones definidas por un sistema del que es usuario legítimo deben ser accesibles. Es posible ver Latch como un factor de autorización más que uno de autenticación, que permite reducir la exposición que tienen los sistemas o servicios a ser atacados. De alguna manera, reduce el impacto que podrían tener los ataques que supongan el compromiso de las credenciales asociadas a una identidad. Es similar a poner un cerrojo sobre el uso de los recursos en función del uso que hace o espera hacer el usuario legítimo de ellos.

La implementación de un segundo factor de autorización a través de herramientas como Latch hace aumentar el nivel de robustez de una identidad digital en Wordpress. La inclusión de un Plugin en Wordpress de Latch permite proteger y detectar robos de identidades, cuando ésta se intenta utilizar por parte del atacante. Este tipo de protección se ofrece a nivel de usuario, es decir, cada usuario será responsable de la utilización del segundo factor.

El proceso de login de Wordpress permite comprobar si la contraseña que el usuario introduce corresponde realmente a dicho usuario (Mecanismo de autenticación basado en "algo que el usuario conoce"). En el caso de que esto sea así, Wordpress autoriza el acceso a los recursos de acuerdo a su política de autorización. En la decisión de conceder acceso a los recursos se puede añadir a Latch, como factor de autorización. Con la instalación del Plugin de Latch para Wordpress, éste será el encargado de preguntar a Latch por el estado del cerrojo. En el caso de que el cerrojo esté abierto se autoriza a que el usuario acceda a los recursos propios. Si por el contrario, el cerrojo está cerrado se deniega el acceso a los recursos. Además, en el caso negativo, una notificación llegará al usuario que está emparejado con la cuenta, para que en todo momento conozca que su identidad digital se ha intentado utilizar.

2.1. Instalación del Plugin de Latch para Wordpress

Para instalar el Plugin de Latch para Wordpress con el objetivo de fortificar el proceso de login y minimizar el tiempo de exposición de la identidad hay que llevar a cabo una serie de pasos:

- En el fichero php.ini se debe descomentar la línea "extension=curl.so".
- La versión de Wordpress mínima debe ser la 1.5.
- Se debe crear una App en el área de desarrolladores de Latch, en la dirección URL <https://latch.elevenpaths.com>.

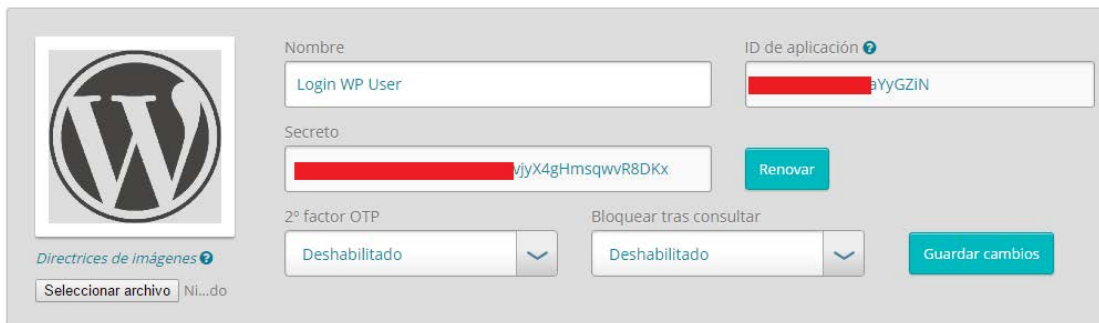


Figura 1: Creación de App en Latch

2.2. Pairing del Plugin de Latch para Wordpress

El proceso de *Pairing* de un usuario con el Plugin de Latch en Wordpress es un proceso sencillo. Desde el perfil del usuario se puede editar cualquier atributo. Entre estos campos se puede encontrar la etiqueta “*Latch Setup*”.

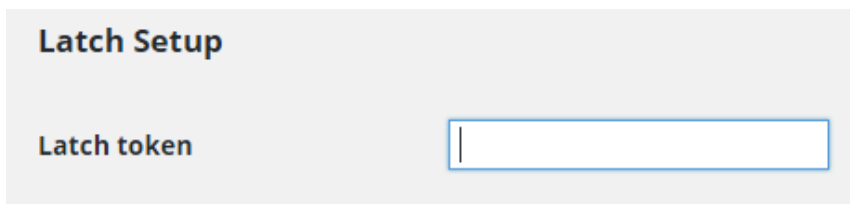


Figura 2: Inserción de token de Latch en Wordpress

Tras generar un *token* con Latch e introducir en el cuadro de texto anterior, se ha llevado a cabo el proceso de *Pairing*. Una notificación se encuentra en su aplicación de Latch para indicar la protección generada.

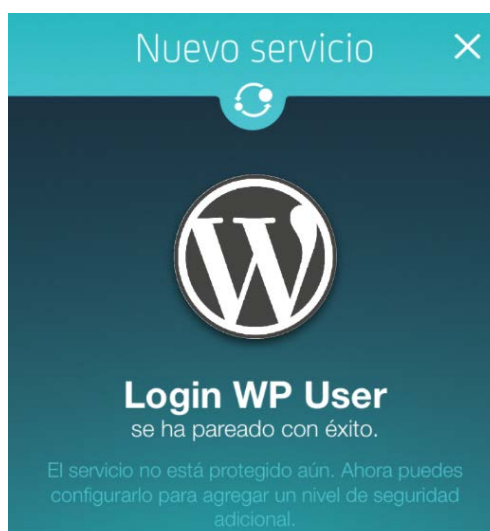


Figura 3: Pairing de Latch con Plugin de Wordpress

2.3. Verificación ante intento de suplantación de identidad en Wordpress

Para verificar que el usuario tiene su identidad protegida ante la pérdida o robo de credenciales se debe habilitar el cerrojo, tal y como se puede ver en la imagen.

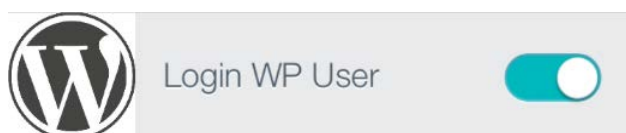


Figura 4: Latch bloqueado

En el caso de un intento de suplantación de identidad, Wordpress comprobará en primer lugar que el usuario y contraseña es correcto. En tal caso, se realizará una consulta contra Latch para comprobar el estado del cerrojo. Si el cerrojo se encuentra habilitado, Wordpress denegará el acceso al usuario y Latch generará una notificación para alertar al usuario de un potencial intento de suplantación de identidad. En el caso de que el cerrojo se encontrase abierto, es decir, deshabilitado, se haría el *login* de forma normal, dando acceso al usuario a la zona interna de Wordpress.

3. Fortificación a nivel de tabla con 2 Factor Authorization

Cuando ataques como SQL Injection o Network Packet Manipulation impactan en el motor de base de datos de Wordpress se puede provocar la pérdida, alteración o inserción de información. Un SQL Injection puede provocar la pérdida, la manipulación o la inserción de datos en tablas críticas de Wordpress, como podrían ser *"wp_users"* o *"wp_usermeta"*. Por otro lado, un ataque de Network Packet Manipulation permite a un atacante, que se ha colocado en medio de la comunicación entre la aplicación web de Wordpress y la base de datos, modificar la query que el CMS realiza la base de datos. El objetivo del atacante será alterar, eliminar o insertar información. Se entiende que estas dos amenazas son importantes y que afectan a la confidencialidad e integridad de forma directa, y a la disponibilidad de forma indirecta.

La solución propuesta, y que se ha llevado a cabo, es la implementación de una serie de *triggers* que monitorizan las acciones que ocurren sobre las tablas críticas de Wordpress, permitiendo al propietario habilitar o deshabilitar, mediante un segundo factor de autorización, acciones sobre las tablas de Wordpress. Estos *triggers* se comunicarán con Latch para comprobar el estado de la operación a llevar a cabo. En caso de que la operación se encuentre con el cerrojo cerrado, el *trigger* bloqueará la *query*, mientras que en caso de que la operación tenga el cerrojo abierto, el *trigger* permitirá la acción primaria.

3.1. Modo de actuación

Para entender mejor el modelo de seguridad propuesto basado en la fortificación a nivel de tabla de un CMS se exponen los siguientes modos de actuación:

- Modo *"Read-Only"*. El modo sólo lectura permite al propietario del Wordpress deshabilitar la posibilidad de logarse de un usuario, así como operaciones administrativas como la adición, modificación o eliminación de usuarios. Tampoco se puede publicar ningún tipo de comentario, ni de post en el CMS.

- Modo *“Administration”*. El modo de administración permite al propietario del Wordpress decidir cuándo se aceptan inserciones, actualizaciones o eliminaciones de usuarios, es decir, gestionar usuario cuando el propietario requiera.
- Modo *“Edition”*. El modo de edición permite al propietario del Wordpress delegar la posibilidad de publicar contenidos, sólo cuando él lo requiera.

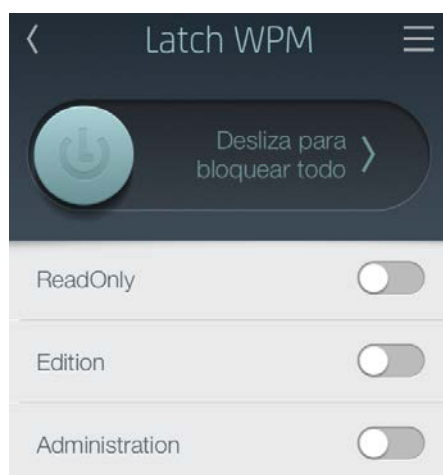


Figura 5: Modos de operación con Latch WPM

Estos modos responden a una protección clara, solamente se podrán realizar ciertas operaciones críticas cuando el propietario habilite la posibilidad mediante un segundo factor de autorización. El primer factor de autorización serían los permisos de la propia base de datos.

Se puede entender que ataques de SQL Injection o Network Packet Manipulation quedaran protegidos contra la inserción, modificación y eliminación de contenidos. El elemento de monitorización de tablas son los triggers, los cuales se encuentran en los motores de bases de datos y permiten ejecutar código. Esta ejecución de código sirve de segundo factor de autorización para comprobar si llevar a cabo una acción u otra.

3.2. Tablas afectadas

Una instalación de Wordpress por defecto crea 12 tablas en su motor de base de datos, generalmente MySQL. En la siguiente imagen se puede visualizar las diferentes tablas que son creadas en una instalación de Wordpress.

```

+-----+
| Tables_in wordpress |
+-----+
| wp_commentmeta      |
| wp_comments         |
| wp_links            |
| wp_options          |
| wp_postmeta         |
| wp_posts            |
| wp_term_relationships |
| wp_term_taxonomy    |
| wp_termmeta         |
| wp_terms            |
| wp_usermeta         |
| wp_users            |
+-----+
  
```

Figura 6: Tablas por defecto en Wordpress

Según el análisis realizado para cumplir con los diferentes modos propuestos en el apartado anterior se debe crear *triggers* en las tablas *wp_comments*, *wp_posts*, *wp_users* y *wp_usermeta*.

La tabla *wp_comments* almacena los comentarios y los relaciona con los *posts*. Además, existen una serie de metadatos, como son el autor, email, dirección URL, etcétera. La idea es que cuando el modo “*Read-Only*” esté habilitado se proteja la posibilidad de poder insertar, actualizad o eliminar contenido de esta tabla, por lo que no se podría crear nuevos comentarios.

Field	Type	Null	Key	Default	Extra
comment_ID	bigint(20) unsigned	NO	PRI	NULL	auto_increment
comment_post_ID	bigint(20) unsigned	NO	MUL	0	
comment_author	tinytext	NO		NULL	
comment_author_email	varchar(100)	NO	MUL		
comment_author_url	varchar(200)	NO			
comment_author_IP	varchar(100)	NO			
comment_date	datetime	NO		0000-00-00 00:00:00	
comment_date_gmt	datetime	NO	MUL	0000-00-00 00:00:00	
comment_content	text	NO		NULL	
comment_karma	int(11)	NO		0	
comment_approved	varchar(20)	NO	MUL	1	
comment_agent	varchar(255)	NO			
comment_type	varchar(20)	NO			
comment_parent	bigint(20) unsigned	NO	MUL	0	
user_id	bigint(20) unsigned	NO		0	

Figura 7: Campos de la tabla *wp_comments*

La tabla *wp_posts* contiene diferentes atributos relacionados con la identidad del autor del post, la fecha, la visibilidad, etcétera. El modo edición se encargará de proteger las inserciones, actualizaciones o eliminaciones de valores de la tabla. Además, si el modo “*Read-Only*” está activo no se puede realizar la manipulación de la tabla.

Field	Type	Null	Key	Default	Extra
ID	bigint(20) unsigned	NO	PRI	NULL	auto_increment
post_author	bigint(20) unsigned	NO	MUL	0	
post_date	datetime	NO		0000-00-00 00:00:00	
post_date_gmt	datetime	NO		0000-00-00 00:00:00	
post_content	longtext	NO		NULL	
post_title	text	NO		NULL	
post_excerpt	text	NO		NULL	
post_status	varchar(20)	NO		publish	
comment_status	varchar(20)	NO		open	
ping_status	varchar(20)	NO		open	
post_password	varchar(20)	NO			
post_name	varchar(200)	NO	MUL		
to_ping	text	NO		NULL	
pinged	text	NO		NULL	
post_modified	datetime	NO		0000-00-00 00:00:00	
post_modified_gmt	datetime	NO		0000-00-00 00:00:00	
post_content_filtered	longtext	NO		NULL	
post_parent	bigint(20) unsigned	NO	MUL	0	
guid	varchar(255)	NO			
menu_order	int(11)	NO		0	
post_type	varchar(20)	NO	MUL	post	
post_mime_type	varchar(100)	NO			
comment_count	bigint(20)	NO		0	

Figura 8: Campos de la table *wp_posts*

La tabla *wp_users* y *wp_usermeta* están fuertemente relacionada. En la primera tabla se almacena información sobre los usuarios, así como su contraseña, email, dirección URL, nickname, etcétera. Esta tabla es utilizada por el CMS para

poder comparar si la contraseña introducida por el usuario es válida y se corresponde con el nombre de usuario indicado.

Field	Type	Null	Key	Default	Extra
ID	bigint(20) unsigned	NO	PRI	NULL	auto_increment
user_login	varchar(60)	NO	MUL		
user_pass	varchar(255)	NO			
user_nicename	varchar(50)	NO	MUL		
user_email	varchar(100)	NO	MUL		
user_url	varchar(100)	NO			
user_registered	datetime	NO		0000-00-00 00:00:00	
user_activation_key	varchar(255)	NO			
user_status	int(11)	NO		0	
display_name	varchar(250)	NO			

Figura 9: Campos de la tabla *wp_users*

Cuando el usuario es validado se crea una cookie de sesión la cual se almacena en *wp_usermeta*. La estructura de la tabla *wp_usermeta* se muestra a continuación.

Field	Type	Null	Key	Default	Extra
umeta_id	bigint(20) unsigned	NO	PRI	NULL	auto_increment
user_id	bigint(20) unsigned	NO	MUL	0	
meta_key	varchar(255)	YES	MUL	NULL	
meta_value	longtext	YES		NULL	

Figura 10: Campos de la tabla *wp_usermeta*

La cookie de sesión se crea como “*session_tokens*” y se almacena en la columna “*meta_key*”, mientras que el valor de esa cookie se almacena en “*meta_value*”. Otro valor importante es el rol, el cual se almacena también en la tabla *wp_usermeta*. Este valor es vital, ya que va asociado a los permisos que el usuario tiene en el CMS.

Esta información es vital a la hora de diseñar los *triggers*. Existen los algunos escenarios críticos que pueden necesitar un tratamiento especial:

- Cuando se elimina, se actualiza o se añade un nuevo usuario se modifica la tabla *wp_users*, pero también la tabla *wp_usermeta*, ya que el rol, *nickname* y otros atributos van relacionados con el usuario.
- Cuando se loguea un usuario hay consultas a la tabla *wp_users* para comprobar que la contraseña es la que se tiene almacenada. Cuando se comprueba y valida se genera una inserción en la tabla *wp_usermeta* para añadir, entre otras cosas, el *session_token*. Si éste existiera, en vez de realizarse un INSERT se haría una operación de UPDATE con el nuevo valor, almacenado en “*meta_value*”.

En el diseño de la solución se ha tenido en cuenta estos escenarios, en el que ambas tablas se encuentran relacionados. Se ha identificado que la tabla que más alteraciones o modificaciones puede sufrir es *wp_usermeta*, por lo que en el siguiente apartado se detalla el algoritmo utilizado para la implementar la solución.

3.3. Esquema global de WPM

La solución propuesta para proteger Wordpress a nivel de tabla con Latch tiene diferentes elementos. A continuación se muestran los elementos de los que consta la solución:

- Binarios que implementan las consultas a Latch, según la operación que se necesite consultar.
- Binario denominado *token* para *pairing* con Latch.
- Binario denominado *operations* que se encarga de crear las operaciones una vez el *pairing* está realizado satisfactoriamente.
- *Triggers* generados que interactúan con los binarios encargados de consultar el estado de la operación de Latch.

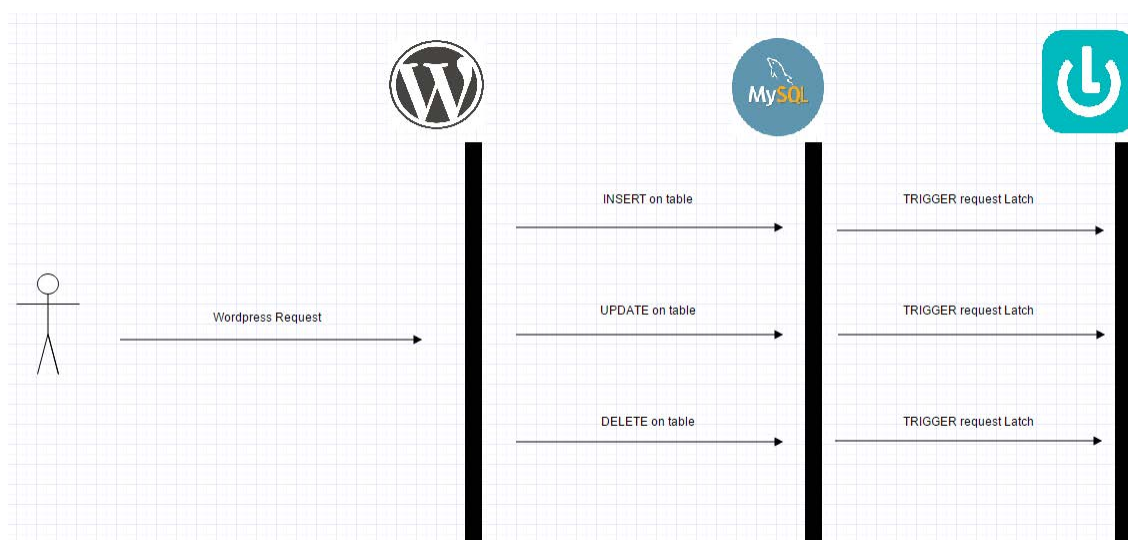


Figura 11: Esquema WPM

3.4. Algoritmo de los triggers

Los *triggers* asociados a las tablas son idénticos en sus tres posibles operaciones (INSERT, UPDATE, DELETE). Entre tablas pueden cambiar ligeramente su algoritmo. En este apartado se muestran diferentes ejemplos de los triggers diseñados e implementados como parte central de protección en *Wordpress in Paranoid Mode*. El algoritmo genérico para las operaciones en pseudocódigo es el siguiente:

CREAR TRIGGER nombreTrigger

ANTES/DESPUES en baseDatos.tabla

PARA CADA FILA

Readonly = Ejecutar y almacenar estado operación "Read-Only" a Latch

SI Readonly != 0 ENTONCES

Bloquear modificación en la tabla "modo sólo lectura"

FIN SI

Operación = Ejecutar y almacenar estado operación X a Latch

SI Operación != 0 ENTONCES

Bloquear modificación en la tabla "modo X"

FIN SI

FIN TRIGGER

El caso de la tabla wp_usermeta presenta un escenario crítico, presentado anteriormente, y es que cuando se inicia sesión se necesita poder escribir o modificar un valor en dicha tabla. Si el modo Administration estuviera activo no se podría realizar el inicio de sesión. Para solucionar esto se diseñó el siguiente algoritmo para los triggers de la tabla wp_usermeta.

CREAR TRIGGER nombreTrigger

ANTES/DESPUES en baseDatos.tabla

PARA CADA FILA

Readonly = Ejecutar y almacenar estado operación "Read-Only" a Latch

SI Readonly != 0 ENTONCES

Bloquear modificación en la tabla "modo sólo lectura"

FIN SI

SI NEW.meta_key != 'session_tokens' ENTONCES

Admin = Ejecutar y almacenar estado operación "Administration" a Latch

SI Admin != 0 ENTONCES

Bloquear modificación en la tabla "modo Administration"

FIN SI

FIN_SI

FIN TRIGGER

Para ejemplificar esto con implementación se muestra el siguiente ejemplo sobre el *trigger* que se ejecutaría ante una instrucción de inserción en la tabla *wp_usermeta*.

CREATE TRIGGER LatchUsermetaUpdateWP

BEFORE UPDATE ON wordpress.wp_usermeta

FOR EACH ROW

BEGIN

DECLARE readonly int;

DECLARE result int;

SET readonly = sys_exec('ruby %PATH%/comment.rb');

```

    IF readonly NOT IN (0) THEN

        SIGNAL SQLSTATE '45000'

        SET MESSAGE_TEXT = 'Latch Cerrado';

    END IF;

    IF NEW.meta_key <> 'session_tokens' THEN

        SET result = sys_exec('ruby %PATH%/users.rb ');

        IF result NOT IN (0) THEN

            SIGNAL SQLSTATE '45000' -- "unhandled user-defined exception"

            SET MESSAGE_TEXT = 'Latch Cerrado';

        END IF;

    END IF;

END

```

3.5. Proceso de instalación

El proceso de instalación se ha automatizado completamente a través de la consola BASH. Este proceso se divide en diferentes pasos, los cuales se enumeran a continuación:

- Se debe crear una App de Latch en el área de desarrolladores. Los valores *Application ID* y *Secret* deben ser tenidos en cuenta para pasárselos al instalador de WPM.

Mis aplicaciones

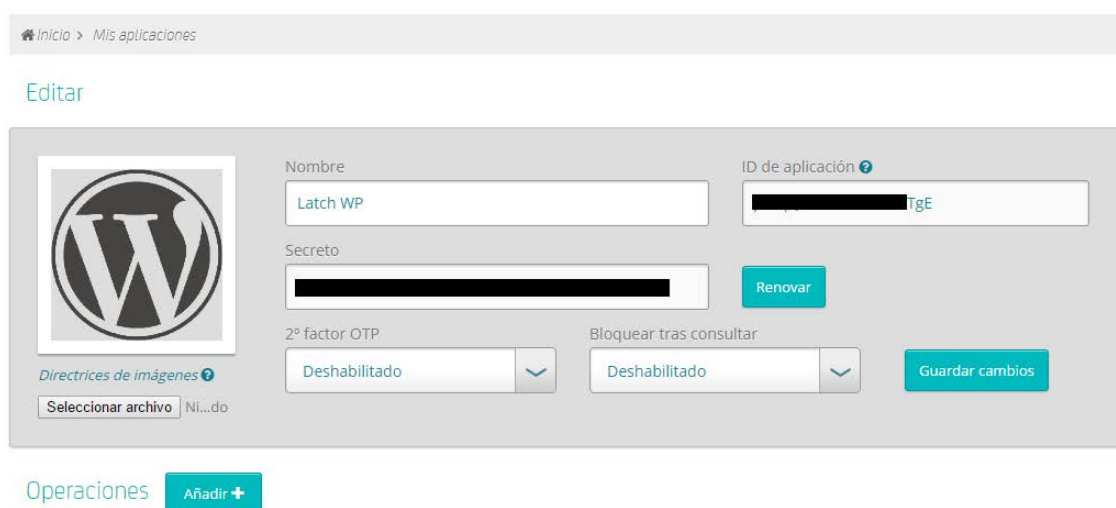


Figura 12: Creación de App en Latch

- *Pairing* con Latch. A través de la solicitud de un *token* a Latch se realiza el proceso de *pairing*.

```

Go to Install? ENTER...

Step 1: Pairing with Latch
=====
Give me token:8xqbh2
Pairing...
Account ID: frrUxzjysk JrNVLmwMet2WYZ:
  
```

Figura 13: Pairing de WPM

- Una vez se realiza el proceso de *Pairing*, se lleva a cabo el proceso de copia y personalización de archivos con los datos de *APP ID*, *SECRET*, *ACCOUNT ID*. Posteriormente, se realiza la creación de las operaciones: “*Read-Only*”, “*Administration*” y “*Edition*” en Latch. Además, los ficheros de los binarios que interactúan con Latch se personalizan con los distintos *OPERATION ID*.
- En este punto, todo lo necesario para interactuar con Latch está creado y configurado. Se instalarán las dependencias y aplicaciones necesarias, así como se configura el motor de base de datos con lo necesario para, posteriormente, realizar el volcado y creación de *triggers*.
- En este último punto se lleva a cabo la creación de los *triggers* en el motor de base de datos sobre las tablas especificadas anteriormente.

```

Step 6: Creating Triggers on MySQL
=====
Enter password:
Success! Triggers on MySQL
  
```

Figura 14: Finalización correcta de instalación de WPM

3.6. Protegiendo Wordpress ante acciones maliciosas

El modo “*Read-Only*” configura Wordpress para que no se puede escribir en él, por lo que tampoco se podrá llevar a cabo el *login* de ningún usuario. Como se puede visualizar en la imagen, se cierra la operación “*Read-Only*” en la App de Latch.

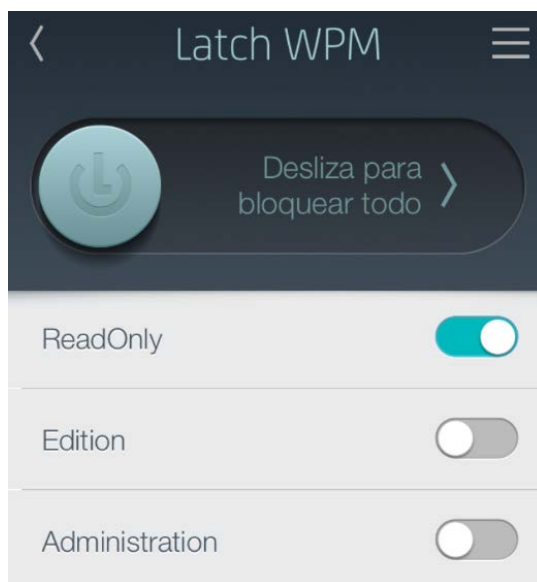


Figura 15: Modo Read-Only activo

En el instante en el que se intenta escribir un comentario en Wordpress (no se necesita estar logar, y podría ser cualquier usuario de Internet) o cualquier usuario intenta iniciar sesión, los *triggers* comprobarán el estado de Latch para dicha operación. En ese instante se abortarán las inserciones, actualizaciones o eliminaciones que se intenten en las tablas. Además, el usuario propietario del Wordpress recibirá una notificación de intento de acceso. Esto quiere decir que se ha realizado una operación contra alguna tabla, y que el *trigger* asociado lo ha bloqueado.

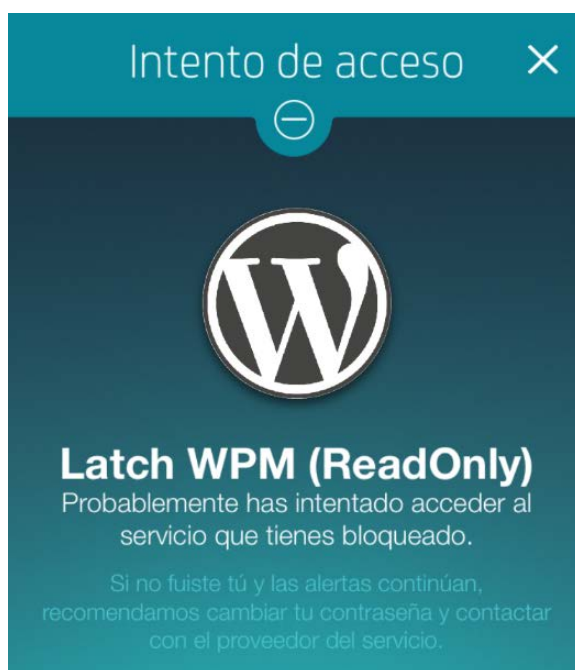


Figura 16: Intento de acceso de cualquier usuario con el modo Read-Only activo

En el caso del modo “*Administration*” si se intenta eliminar, actualizar o insertar valores a la tabla de *wp_users*, y de forma indirecta en *wp_usermeta*, se bloqueará, siempre y cuando la operación *Administration* se encuentra cerrada. En la siguiente imagen se puede visualizar cómo se intenta eliminar un usuario.

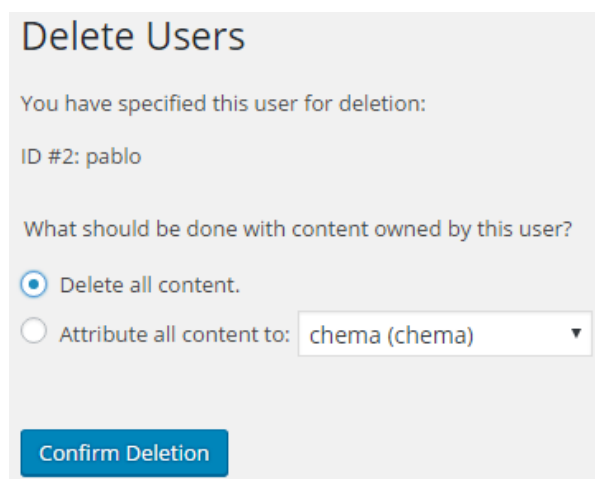


Figura 17: Eliminar usuarios en Wordpress

Si la operación está abierta, se dejará llevar a cabo la eliminación del usuario, pero si el propietario de Wordpress tiene el cerrojo echado en Latch sobre esa operación se bloqueará dicho *DELETE* en la tabla. En la imagen se puede visualizar cómo se notifica el intento de acción administrativa.

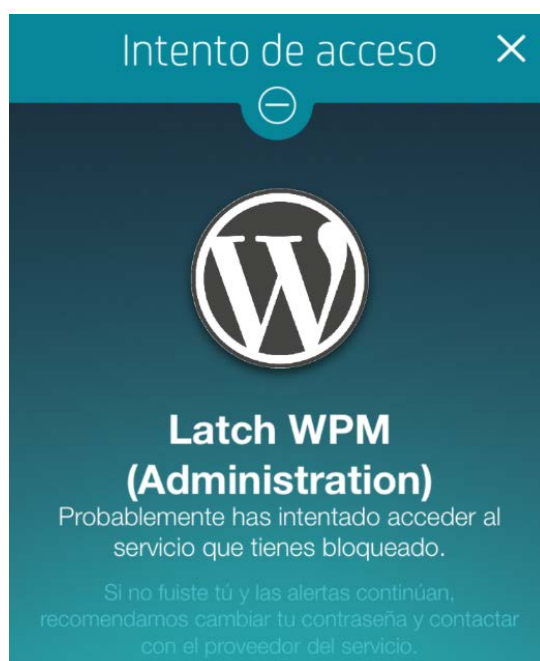


Figura 18: Intento de eliminar usuario bloqueado por el modo Administration

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

2016 © Telefónica Digital España, S.L.U. Todos los derechos reservados.

La información contenida en el presente documento es propiedad de Telefónica Digital España, S.L.U. ("TDE") y/o de cualquier otra entidad dentro del Grupo Telefónica o sus licenciantes. TDE y/o cualquier compañía del Grupo Telefónica o los licenciantes de TDE se reservan todos los derechos de propiedad industrial e intelectual (incluida cualquier patente o copyright) que se deriven o recaigan sobre este documento, incluidos los derechos de diseño, producción, reproducción, uso y venta del mismo, salvo en el supuesto de que dichos derechos sean expresamente conferidos a terceros por escrito. La información contenida en el presente documento podrá ser objeto de modificación en cualquier momento sin necesidad de previo aviso.

La información contenida en el presente documento no podrá ser ni parcial ni totalmente copiada, distribuida, adaptada o reproducida en ningún soporte sin que medie el previo consentimiento por escrito por parte de TDE.

El presente documento tiene como único objetivo servir de soporte a su lector en el uso del producto o servicio descrito en el mismo. El lector se compromete y queda obligado a usar la información contenida en el mismo para su propio uso y no para ningún otro.

TDE no será responsable de ninguna pérdida o daño que se derive del uso de la información contenida en el presente documento o de cualquier error u omisión del documento o por el uso incorrecto del servicio o producto. El uso del producto o servicio descrito en el presente documento se regulará de acuerdo con lo establecido en los términos y condiciones aceptados por el usuario del mismo para su uso.

TDE y sus marcas (así como cualquier marca perteneciente al Grupo Telefónica) son marcas registradas. TDE y sus filiales se reservan todos los derechos sobre las mismas.