Guia de instalacion de Kali VM:

<http://www.cursodehackers.com/VirtualBox.html>

Docker de WebGoat

- Ya visto en la practica, solo es arrancarlo

Ataque de diccionario - video

<https://www.youtube.com/watch?v=elYohMqGH3c>

Repositorio de GitHub de donde he sacado mi (primer) diccionario:

<https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10-million-password-list-top-1000000.txt>

Acceder desde Kali al WebGoat del localhost:

<https://stackoverflow.com/questions/1261975/addressing-localhost-from-a-virtualbox-virtual-machine?rq=1>

macOS

I'm running Virtual Box on macOS (previously OS X), using Virtual Box to test IE on Windows, etc.

Go to IE in Virtual Box and access localhost via http://10.0.2.2 for localhost, or http://10.0.2.2:3000 for localhost:3000.

I kept Network settings as NAT, no need for bridge as suggested above in my case. There is no need to edit any config files.

Tl;dr: http://10.0.2.2:3000

Ejecutar ataque de fuerza bruta de Metasploit:

use scanner/http/wordpress\_login\_enum

set PASS\_FILE /root/Desktop/”archive de passwords”

set

=> Esto es para Wordpress

Asi que:

- Primer ataque: acceso a Wordpress => Container de Wordpress

- Ataques XSS, SQLi, (wtc) => Container de WebGost / DVWA

O si ya está creado, pero parado:

docker start tfm\_wordpress\_1

docker start tfm\_phpmyadmin\_1

docker start tfm\_db\_1

Codigo para crear el Dockerfile adecuado de Wordpress:

<https://gist.github.com/bradtraversy/faa8de544c62eef3f31de406982f1d42>

(video: <https://www.youtube.com/watch?v=pYhLEV-sRpY>)

Desde Kali se puede ver la página de wordpress en local:

10.0.2.2:8000

También hay acceso a la base de datos:

10.0.2.2:8080

(esto en local es 10.0.2.2:8080)

Ataques desde Kali > Metasploit:

msfconsole

> use scanner/http/wordpress\_login\_enum

> set PASS\_FILE /root/Desktop/”archive\_de\_passwords.txt”

> set RHOSTS 10.0.2.2

> set RPORT 8000

(> set USERNAME wordpress)

(sacado, más o menos, de aquí: <https://www.youtube.com/watch?v=elYohMqGH3c>)

1. RESUMEN

2. ESTADO DEL ARTE

- Machine Learning

- Ciberseguridad

- Detección de intrusiones

- Machine Learning para detección de intrusiones

3. VISIÓN GENERAL DE LA ARQUITECTURA DEL SISTEMA PROPUESTO

4. ENTORNO DEL ATAQUE

4.1. Definición y objetivos

4.2. Arquitectura detallada

4.2. Sistema atacante

- Máquina virtual

- Kali Linux

- Metasploit

4.3. Sistema vulnerable

- Docker

4.3.1. Aplicación Web Wordpress

- Gestor de logs

- Activity log

- Ataque de diccionario

- Ataque de fuerza bruta

4.3.2. DVWA

- SQLi

- XSS

4.3.3.

4.4. Resultados

- Aquí tenemos los datasets formados por los logs

5. ENTORNO DEL ANÁLISIS DE LA INFORMACIÓN

5.1. Definición y objetivos

5.2. Arquitectura detallada

- Script de Python

5.3. Modelos propuestos

5.3.1. blablá

5.3.2. blablablá

5.3.3. Red neuronal

5.4. Resultados de los modelos

6. ENTORNO DE LA VISUALIZACIÓN

6.1. Definición y objetivos

6.2. Arquitectura detallada

6.3. Implementación de la pila ELK

6.3.1. Elastic

6.3.2. Logstash

6.3.3. Kibana

6.4 Resultados

6.5. Visualización

7. MEJORAS PROPUESTAS AL SISTEMA

7.1. Reconocimiento y detección en tiempo real

7.2. Aplicación web para la monitorización de la aplicación web

ANEXOS

Anexo 1. Declaración del Dockerfile

4. ENTORNO DEL ATAQUE

4.1. Definición y objetivos

En esta primera parte del sistema se implementará la simulación de un ataque a una aplicación web. Para ello, se establecerá una máquina atacante, que ejecutará un ataque de diccionario hacia otra máquina que aloja a la aplicación.

El resultado final será la generación de un archivo que recoja los logs de actividad del sistema vulnerable, que psoteriormente será analizado mediante diferentes técnicas de Machine Learning.

4.2. Arquitectura detallada

La arquitectura de esta parte está conformada por:

- Un atacante, desde el cual se llevará a cabo la intrusión a un sistema objetivo.

El atacante será un sistema basado en la distribución de Linux Kali.

Los vectores de ataque se realizarán mediante el framework Metasploit, con el que se puede obtener información acerca de las vulnerabilidades de seguridad, y desde el cual se hará la prueba de intrusión al sistema.

- Una máquina, que aloja a la aplicación web.

Se ha decidido llevar a cabo la aplicación web con un CMS de tipo Wordpress, por ser éste uno de los método más populares para implementar una página en Internet.

En marzo de 2019 era usado por el 33,4 % de todos los sitios en Internet y un 60,3 % de todos los sitios basados en gestores de contenido (CMS) [[1]](#footnote-1)

Por otro lado, este sitio Wordpress tendrá instalado el plugin Activity Log para el guardado y posterior volcado a un archivo de los logs de actividad.

Ambos sistemas se encontrarán el misma red, aunque esto no tiene ninguna relevancia de cara a los resultados finales: se obtendrían las mismas soluciones si la intrusión se realizara desde un sistema ajeno a la red.

4.3. Implementación de la máquina atacante

El atacante será un sistema basado en la distribución de Linux Kali.

Los vectores de ataque se realizarán mediante el framework Metasploit, con el que se puede obtener información acerca de las vulnerabilidades de seguridad, y desde el cual se hará la prueba de intrusión al sistema.

Para la implementación de la máquina atacante se ha usado una imagen de Kali Linux, disponible en la página web, disponible de forma gratuita en su página web <https://www.kali.org>.

La versión elegida ha sido la más reciente en la actual fecha: 2019.3

Para la virtualización de la máquina se ha usado Oracle VM VirtualBox, y se ha realizado la instalación completa de la imagen (es decir, no se ha elegido la opción LiveCD)

-- Especificaciones de la máquina virtual.

Una vez instalada, se accede con usuario: root, contraseña: root

Una de las herramientas más potentes (y también una de las más usadas) es el framework Metasploit.

Para arrancarlo, basta escribir en una terminal:

msfconsole

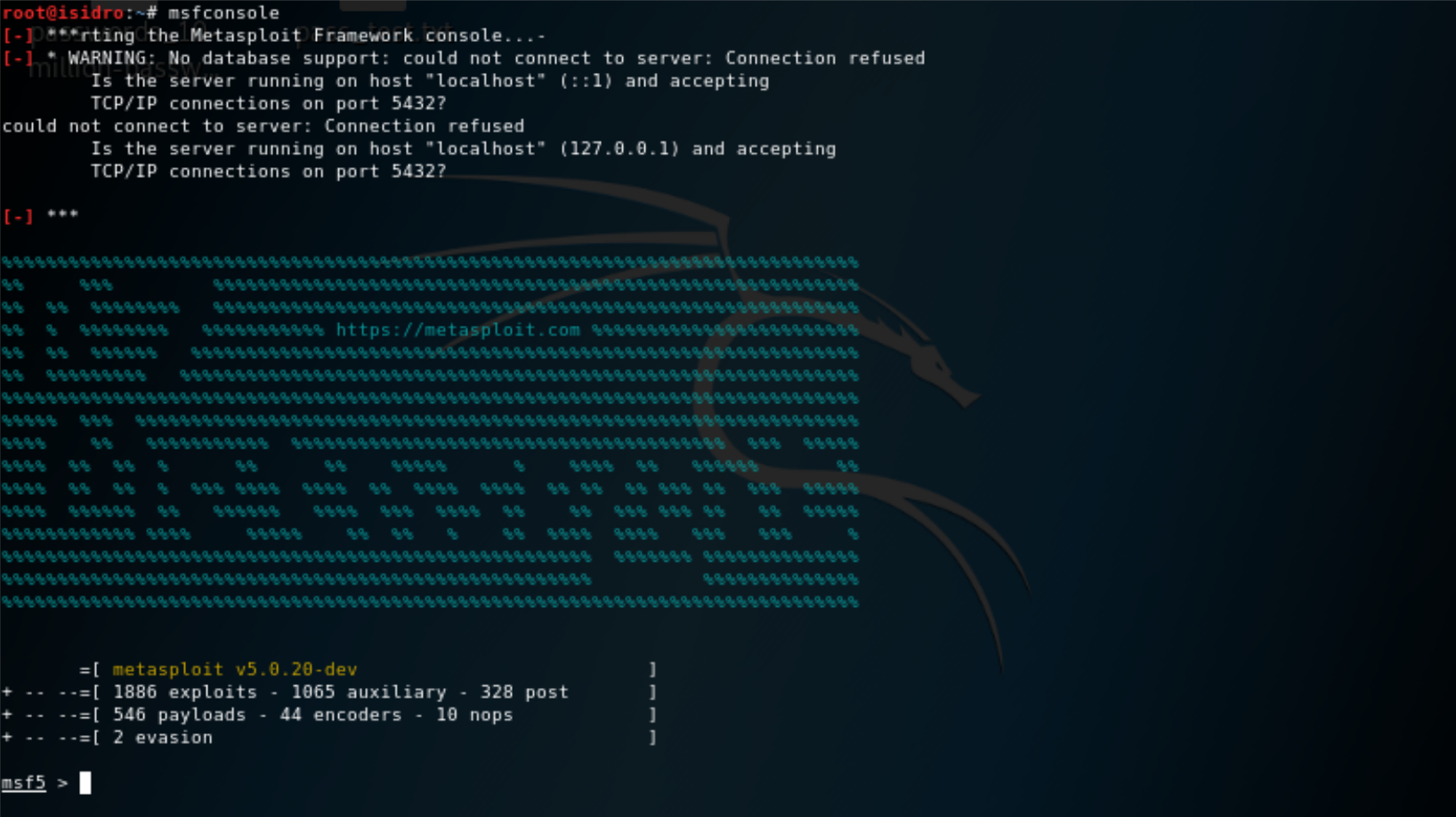


Figura XXXXX. Acceso al framework Metasploit

Como la aplicación a la que se intentará acceder es de tipo Wordpress, podemos usar el módulo auxiliar *wordpress\_login\_enum*, a partir del cual se ejecutará el ataque de diccionario, dado un nombre de usuario válido, tratando de adivinar la contraseña correspondiente.

El módulo *wordpress\_login\_enum* también puede realizar ataques de fuerza bruta.

Para más información, se puede acudir a su sitio en GitHub: https://github.com/rapid7/metasploit-framework/blob/master/documentation/modules/auxiliary/scanner/http/wordpress\_login\_enum.md

Una vez dentro de Metasploit, los comandos que se usarán para realizar el ataque son los siguientes:

> use scanner/http/wordpress\_login\_enum

> set PASS\_FILE /root/Desktop/”archivo\_de\_passwords.txt”

> set RHOSTS 10.0.2.2

> set RPORT 8000

> set USERNAME wordpress

Con el primero de ellos, se seleccionará el módulo *wordpress\_login\_enum*.

El segundo comando permite al módulo de Metasploit elegir un fichero de text que será usado como diccionario para el ataque.

Como se verá más adelante (en el siguiente epígrafe, “4.4. Implementación de la máquina que aloja a la aplicación web”), la máquina que recibirá el ataque se encontrará en la dirección 10.0.2.2:8000. Para indicarlo, usamos el tercer y cuarto comando.

Finalmente, el último comando permite indicar el nombre de usuario para acceder al panel de administración de Wordpress.

En principio, un atacante no tendría porque tener conocimiento del mismo. Sin embargo, para el caso que nos ocupa, se ha decidido que sea así, puesto que de otra forma el ataque supondría un esfuerzo mayor, al incrementarse el número de parámetros a explotar de uno (contraseña) a dos (usuario y contraseña), dando lugar a un tamaño de operaciones N2 (o N x M, en el caso de el tamaño del diccionario para el campo usuario fuera diferente al del diccionario de contraseñas), cuando en el caso de adivinación de contraseña es únicamente N.

Para realizar el ataque, basta escribir el siguiente comando en la terminal de Metasploit:

> exploit

Tras esto, la máquina atacante ya habrá comenzado su actividad.

Se puede monitorizar su avance por las palabras del diccionario en la misma terminal desde donde se ha llevado a cabo el ataque.

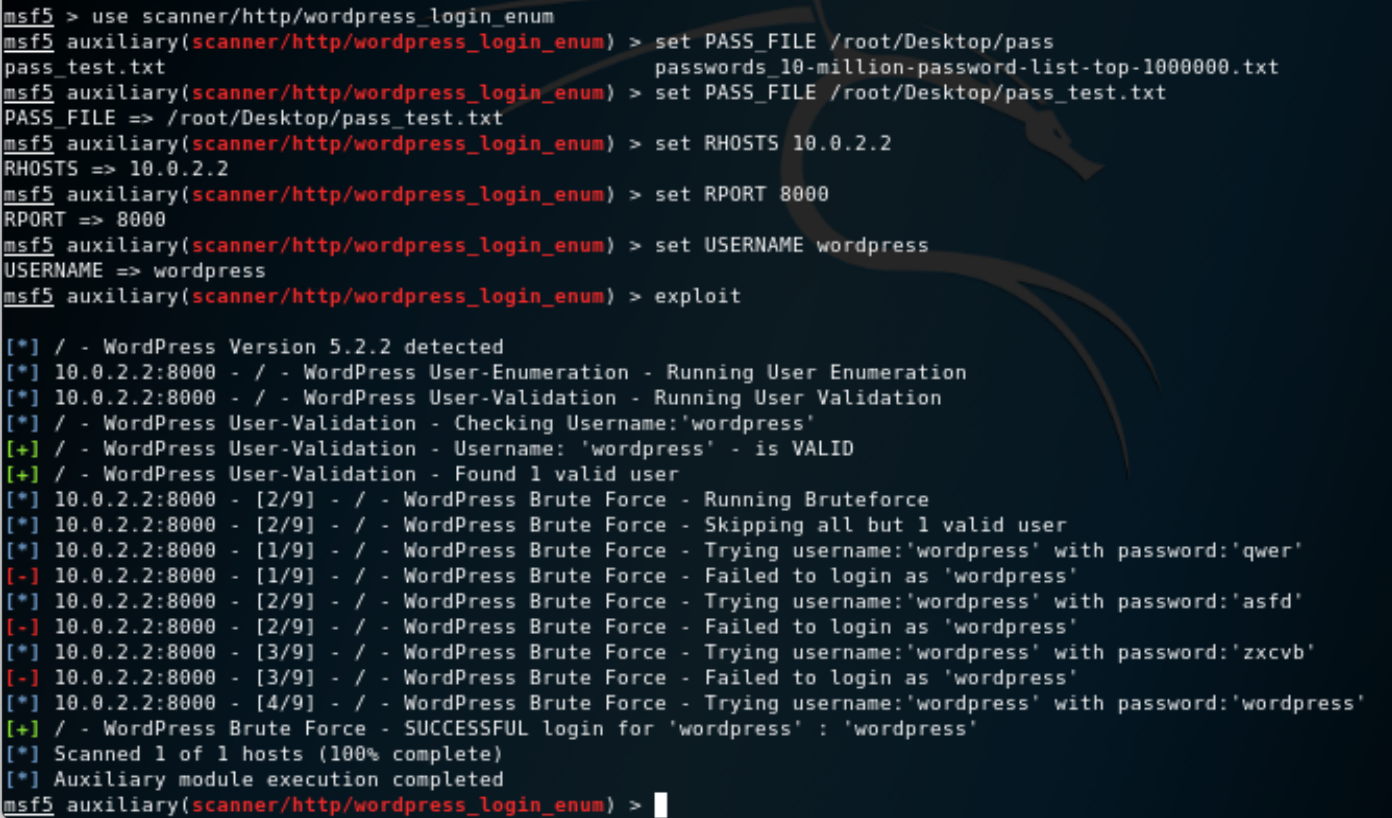


Figura XXXX: Resultado satisfactorio del ataque de diccionario

-- Hablar del diccionario / los diccionarios elegido/s

4.4. Implementación de la máquina que aloja a la aplicación web

En este caso, la implementación del servicio web no se realizará mediante una máquina virtual, sino usando una imagen de Docker.

Docker permite el despliegue de aplicaciones dentro de contenedores independientes, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. De esta forma se evitará la sobrecarga de tener que iniciar y mantener máquinas virtuales.

Para la implementación de la aplicación, se ha codificado un archivo Dockerfile, que permite construir unas imágenes “a medida” mediante la lectura de los comandos indicados en el Dockerfile.

Para la creación de las imágenes, se requiere estar en el directorio del Dockerfile y lanzar el siguiente comando en una terminal:

docker build

El código del docker-compose.yaml usado está especificado en detalle en el apartado de esta memoria: Anexo 1. Declaración del docker-compose.yaml.

De forma breve, podemos indicar que el docker-compose está formado por las siguientes imágenes:

- Wordpress

- mysql

- phpmyadmin

El docker-compose.yaml es un archivo único que permite la instanciación de estos tres módulos, que generarán la versión inciial de la aplicación web.

Cabe indicar que si el servicio ya está creado, pero no activo (es decir, los contenedores se encuentran en modo “stop”), y suponiendo que los nombres de los contenedores de Wordpress, PhpMyAdmin y la base de datos son, respectivamente, *tfm\_wordpress\_1*, *tfm\_phpmyadmin\_1* y *tfm\_db\_1*, los comandos para activarlos son los siguientes:

docker start tfm\_wordpress\_1

docker start tfm\_phpmyadmin\_1

docker start tfm\_db\_1

- Docker

4.4.1. Aplicación Web Wordpress

- Gestor de logs

- a) Activity log

- b) **DOCKER LOGS identificador\_del\_contenedor**

- Ataque de diccionario

- Ataque de fuerza bruta

4.4.2. DVWA

- SQLi

- XSS

4.4.3.

4.5. Conexión entre las dos máquinas

ANEXO 1: Declaración del docker-compose.yaml

Las imágenes que descargará el docker-compose son MySQL, Wordpress y PHPMyAdmin: la primera será usada como base de datos de la aplicación, la segunda es propiamente el servicio o aplicación web, y la tercera permite acceder a la información almacendada en la base de datos.

En las siguientes líneas se puede comprobar las características de cada una:

a) Características de la imagen de MySQL:

- Versión 5.7 de MySQL

- Creación de un volumen, para guardar nuestros datos (“persistencia de la información”). El volumen se mapeará dentro del propio directorio /var/lib/mysql

- Política de reinicio, declarada como “Always”, para que cada vez que el servidor se reinicie, la base de datos también lo haga.

- Tras la declaración del environment, se indican los parámetros de la base de datos.

- El nombre de la red elegido, y que será común para el resto de imágenes usadas en la aplicación web, es “wpsite”.

# Database

  db:

    image: mysql:5.7

    volumes:

      - db\_data:/var/lib/mysql

    restart: always

    environment:

      MYSQL\_ROOT\_PASSWORD: password

      MYSQL\_DATABASE: wordpress

      MYSQL\_USER: wordpress

      MYSQL\_PASSWORD: wordpress

    networks:

      - wpsite

Imagen XXXXX: Código del docker-compose para el servicio MySQL

b) Características de la imagen de Wordpress:

- Dependencia directa de la imagen anterior (“db”)

- Última versión disponible de Wordpress

- Puertos: para la máquina local 8080, mapeado con el puerto del contenedor 80. Cabe indicar que no hace falta establecer una enlace hacia la base de datos, ya que hemos declarado previamente la conexión directa entre ambos y no necesito que la base de datos se exponga hacia el exterior (la máquina local)

- Misma política de reinicio que la base de datos (“Always”)

- Creación de un volumen para mostrar la información del contenedor en la máquina local. Como la imagen de Wordpress usa Apache el directorio root que contiene los archivos para la página web es /var/www/html

- Tras la declaración del environment, se indica la base de datos a la que se conectará (“db”, por el puerto 3306, al ser este el puerto por defecto de MySQL ) y se vuelven a declarar el usuario y contraseña correspondientes, ya declarados en la parte de la base de datos

- La red será la misma que la de la base de datos: “wpsite”

# Wordpress

  wordpress:

    depends\_on:

      - db

    image: wordpress:latest

    ports:

      - '8000:80'

    restart: always

    volumes: ['./:/var/www/html']

    environment:

      WORDPRESS\_DB\_HOST: db:3306

      WORDPRESS\_DB\_USER: wordpress

      WORDPRESS\_DB\_PASSWORD: wordpress

    networks:

      - wpsite

Imagen XXXXX: Código del docker-compose para el servicio Wordpress

c) Características de la imagen de PhpMyAdmin

- Como en el caso de la imagen de Wordpress, ésta también debe depender de la base de datos MySQL

- La imagen de PhpMyAdmin elegida es la que se encuentra en Docker Hub: para ello, indicamos la ruta adecuada desde la cual se descargará (phpmyadmin/phpMyAdmin)

- Misma política de reinicio que la base de datos (“Always”)

- A continuación se indican la conexión entre puertos para poder acceder al servicio desde la máquina local: se hace un mapeo desde el puerto 8080 de la máquina local al puerto 80 del contenedor

- Tras el comando environment, declaramos los parámetros Host y Password (db y password, respectivamente)

- Finalmente indicamos la red, que será la misma que la de los servicios de MySQL y Wordpress

# phpmyadmin

  phpmyadmin:

    depends\_on:

      - db

    image: phpmyadmin/phpmyadmin

    restart: always

    ports:

      - '8080:80'

    environment:

      PMA\_HOST: db

      MYSQL\_ROOT\_PASSWORD: password

    networks:

      - wpsite

Imagen XXXXX: Código del docker-compose para el servicio PhpMyAdmin

En conclusión, el archivo docker-compose.yaml completo es este:

version: '3'

services:

  # Database

  db:

    image: mysql:5.7

    volumes:

      - db\_data:/var/lib/mysql

    restart: always

    environment:

      MYSQL\_ROOT\_PASSWORD: password

      MYSQL\_DATABASE: wordpress

      MYSQL\_USER: wordpress

      MYSQL\_PASSWORD: wordpress

    networks:

      - wpsite

  # phpmyadmin

  phpmyadmin:

    depends\_on:

      - db

    image: phpmyadmin/phpmyadmin

    restart: always

    ports:

      - '8080:80'

    environment:

      PMA\_HOST: db

      MYSQL\_ROOT\_PASSWORD: password

    networks:

      - wpsite

  # Wordpress

  wordpress:

    depends\_on:

      - db

    image: wordpress:latest

    ports:

      - '8000:80'

    restart: always

    volumes: ['./:/var/www/html']

    environment:

      WORDPRESS\_DB\_HOST: db:3306

      WORDPRESS\_DB\_USER: wordpress

      WORDPRESS\_DB\_PASSWORD: wordpress

    networks:

      - wpsite

networks:

  wpsite:

volumes:

  db\_data:

Para ejecutar el archivo docker-compose.yaml, en una terminal se escribe:

docker-compose up -d

Esta instrucción creará la red, los volúmenes, la base de datos y todo lo requerido para iniciar el sitio Wordpress.

El servicio Wordpress se encuentra disponible para su instalación en la dirección localhost:8000/wp-admin/install.php.

Por su parte, el acceso a la base de datos se puede hacer a través de PhpMyAdmin, dirigiéndose a localhost:8080.

DIRECTORIOS IMPORTANTES PARA LOS LOGS DE APACHE

/var/www/html

/var/log

/usr/local/var/log

Notas:

Los contenedores my-own-mysql y my-own-phpmyadmin tienen el proposito de ser usados con Metasploit para hacer un NUEVO TIPO DE ATAQUE remoto, para explotar el PhpMyAdmin (ver enlaces guardados en Chrome/TFM)

=> Importante: la versión de wordpress de esta mierda es la 4.9; y para hacer el ataque hay que usar la 4.8

Los del Wordpress son los de TFM…

Ojo: Se podrían combinar ambos

1. <https://web.archive.org/web/20190315100044/https://w3techs.com/technologies/details/cm-wordpress/all/all> [↑](#footnote-ref-1)