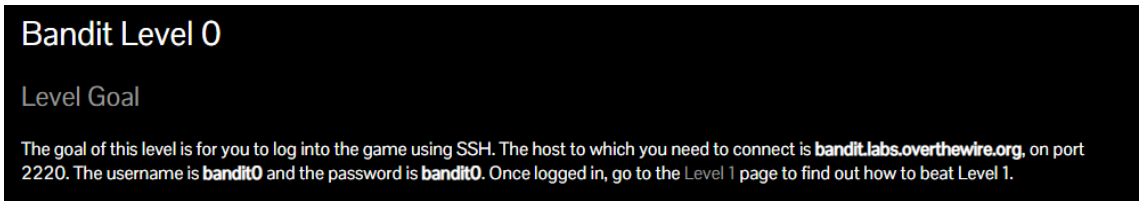
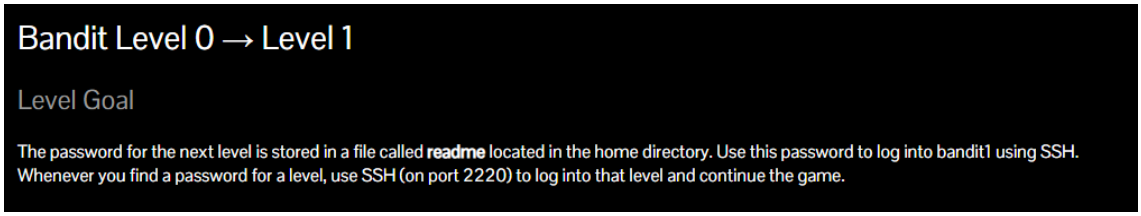
Guía de Niveles Bandit (OverTheWire)

NIVEL 00:



Para conectarnos al bandit, debemos poner el comando: `ssh bandit0@bandit.labs.overthewire.org -p 2220`

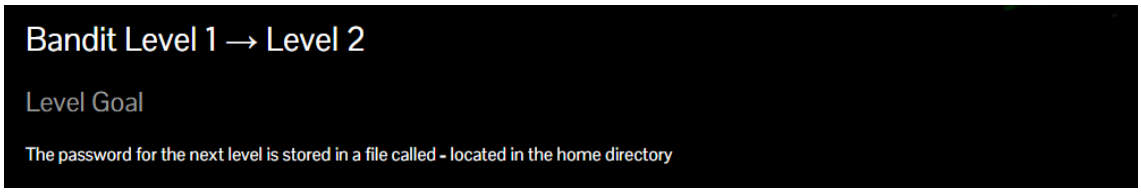
NIVEL 0:



Para comenzar, usamos el comando `pwd` para ver nuestra ubicación actual. Luego listamos los archivos con `ls` y usamos el comando `cat readme` para leer el contenido del archivo. Ahí encontraremos la primera flag que usamos para desbloquear el siguiente nivel:

- `ZjLjTmM6FvvyRnrb2rfNWOZOTa6ip5If`

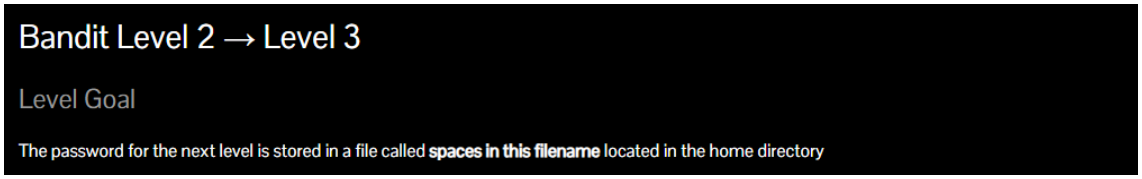
NIVEL 1:



Repetimos los pasos anteriores, pero esta vez nos encontramos con un archivo llamado `-`. Como el guion no se interpreta como nombre de archivo directamente, debemos usar `cat ./-` para leerlo correctamente. Esto nos da la flag:

- `263JGJPfgU6LtdEvgfWU1XP5yac29mFx`

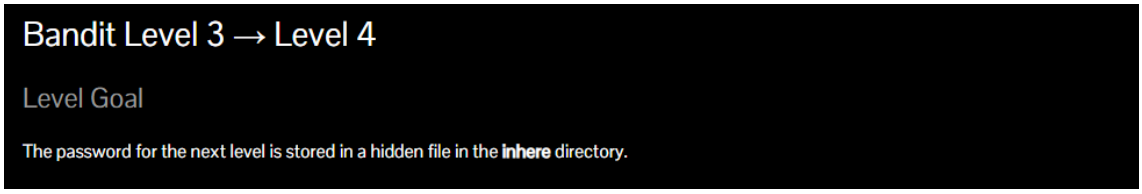
NIVEL 2:



Aquí el archivo tiene espacios en su nombre. Para leerlo, usamos comillas simples: `cat 'spaces in this filename'`. Esto hace que el sistema lo detecte como nombre del archivo y nos da la flag:

- `MNk8KNH3Usiio41PRUEoDFPqfxLPlSmx`

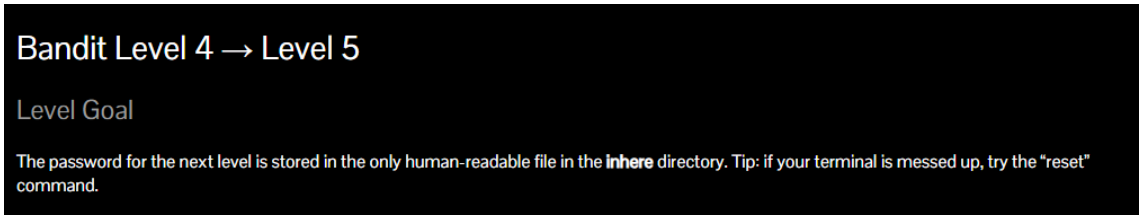
NIVEL 3:



Encontramos un directorio aparentemente vacío. Usamos `ls -la` para ver archivos ocultos y encontramos `..Hiding-From-You`. Lo abrimos con: `cat ...Hiding-From-You` (para abrir archivos ocultos ponemos . antes del nombre del archivo) y obtenemos la flag:

- `2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ`

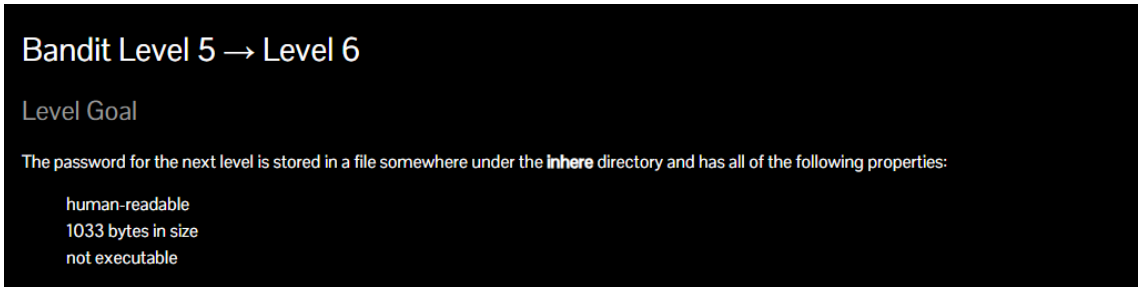
NIVEL 4:



En el directorio `inhere` hay 9 archivos. Solo uno contiene texto legible. Usamos: `file ./*` para saber el tipo de archivo y sobre el archivo de texto (ASCII text) hacemos `cat ./-file07` para obtener la flag:

- `4oQYVPkxZOOEOO5pTW81FB8j8lxXGUQw`

NIVEL 5:



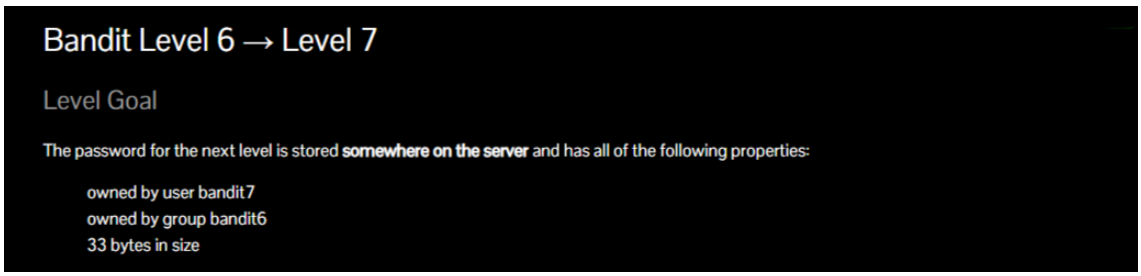
Buscamos en 19 directorios un archivo específico con ciertas características. Ejecutamos:

```
find -type f -size 1033c ! -executable -exec file {} \;
```

Este comando interpreta: `find` **(busca desde el directorio actual),** `-type f` **(solo archivos),** `-size 1033c` **(de 1033 bytes (la c indica "bytes")),** `! -executable` **(que NO sea ejecutable),** `-exec file {} \;` **(muestra el tipo de archivo con el comando file) Una vez lo encontramos, usamos** `cat` **para obtener la flag:**

- `HWasnPhtq9AVKe0dmk45nxy20cvUa6EG`

NIVEL 6:



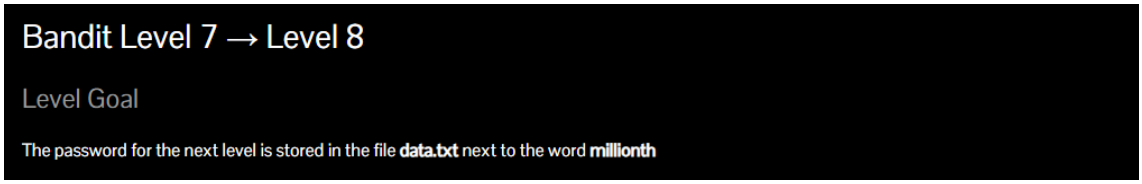
Buscamos un archivo con permisos, sabemos que pertenece a bandit7, al grupo bandit y tiene un tamaño de 33 bytes:

```
find / -type f -user bandit7 -group bandit6 -size 33c 2>/dev/null
```

Este comando interpreta; `/` **(busca desde la raíz),** `-type f` **(solo archivos),** `-user bandit7` **(que pertenezcan a este usuario),** `-group bandit6` **(que pertenezca a este grupo),** `-size 33c` **(tamaño de bytes),** `2>/dev/null` **(oculta los errores de "permiso denegado"). Una vez identificamos el archivo, usamos** `cat` **para ver la flag:**

- `morbNTDkSW6jIlUc0ymOdMaLnOlFVAaj`

NIVEL 7:



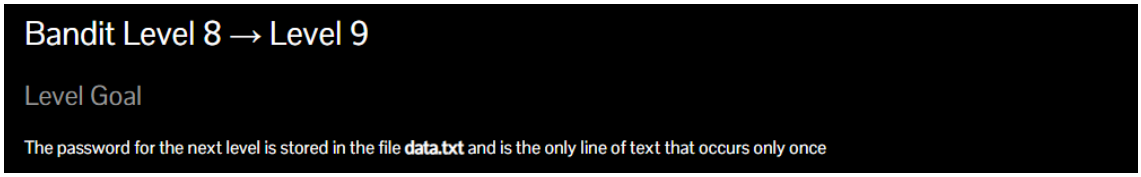
El archivo tiene muchas líneas. Queremos encontrar la línea única que contiene "millionth":

```
sort data.txt | uniq -u | grep 'millionth'
```

Este comando interpreta; `sort data.txt` **(ordena alfabéticamente),** `uniq -u` **(muestra solo las líneas únicas que no están duplicadas),** `grep 'millionth'` **(filtra por la palabra 'millionth'). Esto nos devuelve directamente la flag:**

- `dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc`

NIVEL 8:



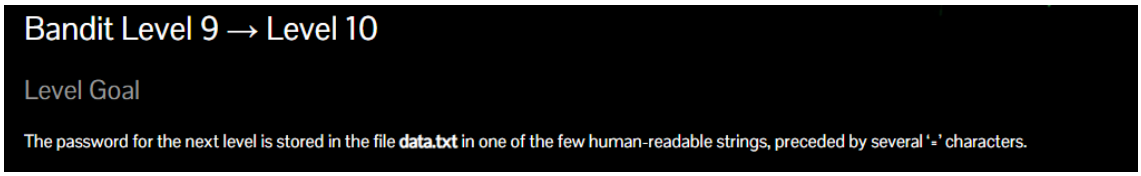
Tenemos un archivo codificado en Base64. Usamos:

```
sort data.txt | uniq -u
```

`sort data.txt` **(nos ordena data.txt),** `uniq -u` **(filtra la cadena unica de texto en el archivo). Esto nos da directamente la flag:**

- `4CKMh1JI91bUIZZPXDqGanal4xvAg0JM`

NIVEL 9:



La contraseña está precedida por signos `=`**. Usamos:**

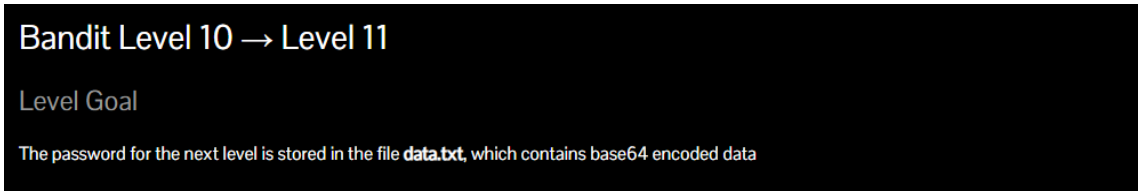
```
strings data.txt | grep '=' | awk -F'=' '{print $NF}'
```

Este comando interpreta: `strings data.txt` **(extrae texto legible del archivo data.txt),** `grep '='` **(filtra por el signo '=' en el texto),** `awk -F'=' '{print $NF}'` **(**`'awk'` **(divide cada línea de texto en campos),** `-`

`F'='`**(el separador es '='),** `$NF`**(significa ultimo campo (el que queremos imprimir)). Y obtenemos la flag:**

- `FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey`

NIVEL 10:



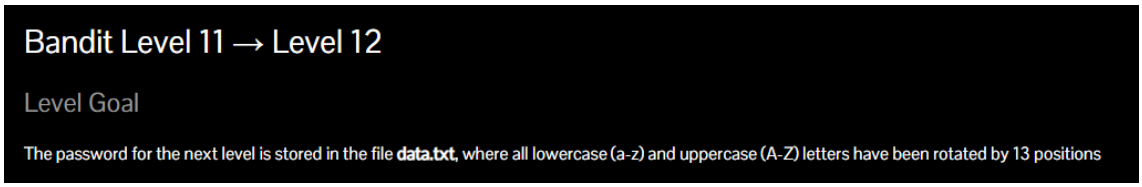
El contenido está en Base64, aparece una línea larga de caracteres y `'=='`**. Decodificamos con:**

```
strings data.txt | base64 -d
```

Y obtenemos la flag:

- `dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr`

NIVEL 11:



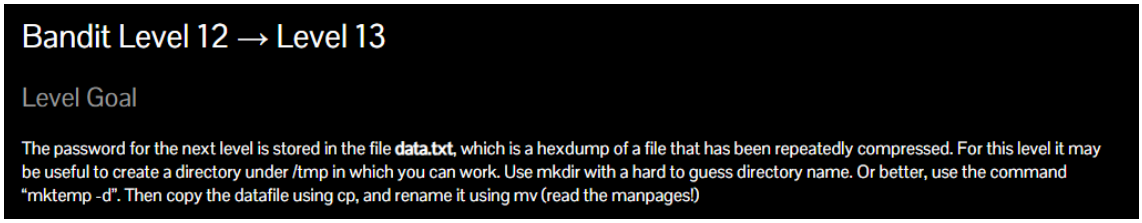
El texto está cifrado con ROT13, es decir, que las letras están rotadas 13 posiciones. Desciframos con:

```
cat data.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m'
```

con el que rotaremos las letras de forma que el texto sea legible y podremos leer la flag:

- `7x16WNeHIi5YkIhWsfFIqoognUTyj9Q4`

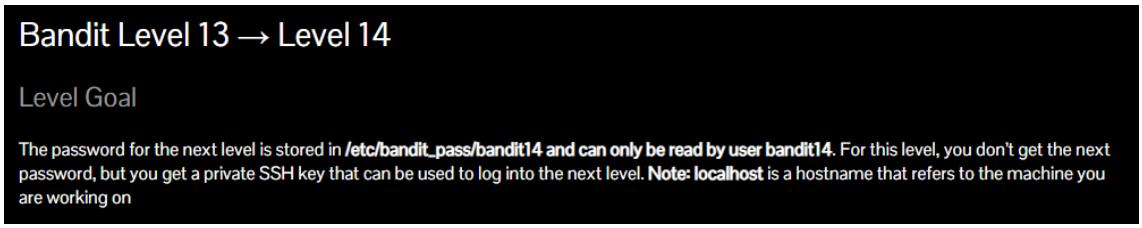
NIVEL 12:



Este archivo está comprimido en múltiples capas. Se debe ir descomprimiendo una a una, para ello debemos; saber que los archivos pueden estar encapsulados en capas: `.gz`, `.bz2`, `.tar`, `.gz`, `.txt`, para saber el tipo de capa en la que estamos, hacemos `file data.txt`. Creamos una carpeta y copiamos el archivo `cp data.txt`, lo renombramos para que sea descomprimible: `mv data.txt compressed_data.gz` y lo descomprimimos: `gzip -d compressed_data.gz` , nos dará otro archivo `compressed_data` y nos dirá que sigue comprimido. Para seguir descomprimiendo, pasamos a la siguiente capa `.bz2` cambiando el nombre del archivo: `mv compressed_data compressed_data.bz2` que descomprimiremos con `bzip2 -d compressed_data.bz2`, luego volvemos a cambiar el nombre del archivo a la siguiente capa: `mv compressed_data compressed_data.gz` y lo descomprimimos `gzip -d compressed_data.gz`, ahora lo volvemos a convertir a la siguiente capa `mv compressed_data compressed_data.tar` y descomprimimos el archivo que nos da `tar -xf data5.bin` ahora descomprimimos el archivo `.bin` que nos ha dado `bzip2 -d data6.bin` y nos dará otro archivo `.out` que descomprimiremos a continuacion `tar -xf data6.bin.out` esto nos dará a su vez otro archivo `.bin` que volveremos a cambiar el nombre y lo descomprimiremos `mv data8.bin data8.gz` y `gzip -d data.gz` y nos dará el archivo final que abriremos con `cat data8`. Finalmente, se obtiene la flag:

- `FO5dwFsc0cbaIiH0h8J2eUks2vdTDwAn`

NIVEL 13:



Intentando acceder a archivo sshkey.private aplicando un cambio de privilegios,

```
chmod 600 sshkey.private
```

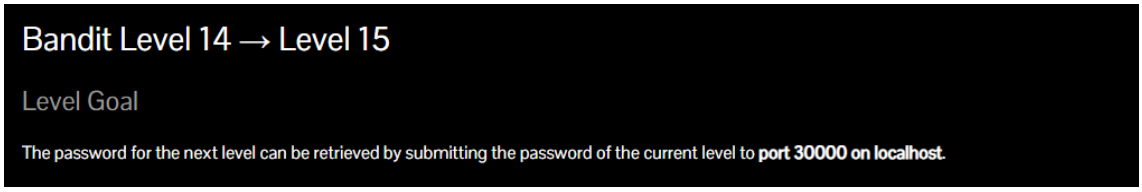
vemos no tenemos permisos para abrir el archivo, por lo tanto vamos a conectarnos al siguiente usuario usando una clave ssh:

```
ssh -i sshkey.private bandit14@localhost -p 2220
```

nos llevará a la máquina virtual de bandit14 donde tendremos que buscar la contradseña del usuario aplicando `"cat /etc/bandit_pass/bandit14"` y nos dará la flag:

- `MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS`

NIVEL 14:



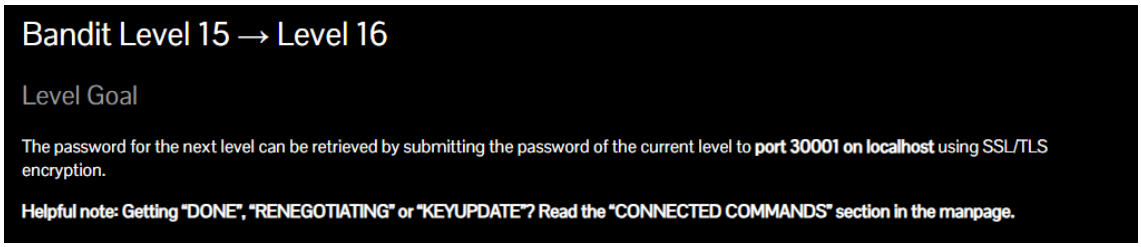
Tenemos que enviar por el puerto `30000` la flag de bandit14 que hemos encontrado en el nivel 13. Para ello desde el sistema de bandit14 aplicamos el siguiente comando:

```
echo MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS | nc localhost 30000
```

y veremos en pantalla la flag:

- `8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo`

NIVEL 15:



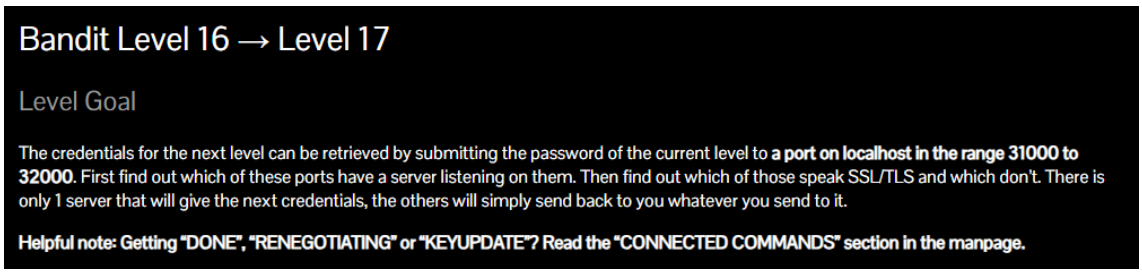
Tenemos que conectarnos a bandit16 y enviar la contraseña del nivel anterior `8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo` por el puerto `30001` para que el usuario bandit16 nos conteste la flag que necesitamos. Usamos openssl para enviar la contraseña:

```
openssl s_client -connect localhost:30001
```

El comando interpreta: `openssl` (herramienta de cifrado de `ssl` que permite conexiones seguras), `s_client` (activa el modo cliente como si fueras un programa que se conecta a un servidor `https`), `localhost` (conecta con el servidor, en este caso la propia máquina), `30001` (puerto por el que se conecta). Pegamos la contraseña anterior y obtenemos la flag:

- `kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx`

NIVEL 16:



La flag se encuentra en un lugar entre los puertos `31000` y `32000` para encontrarla debemos seguir los siguientes pasos, lo primero es saber que puertos están abiertos:

```
nmap -sV localhost -p 31000-32000
```

Lo siguiente que tenemos que hacer será comprobar cual es el puerto correcto, para ello conectaremos con cada puerto abierto con el siguiente comando:

```
openssl s_client -connect localhost:31790
```

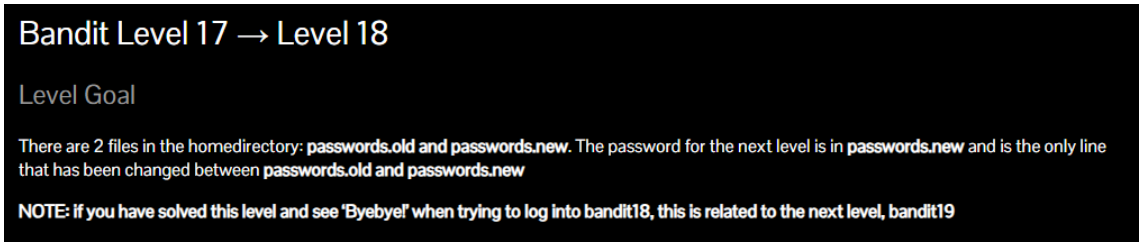
Hasta que nos salga el puerto correcto, lo siguiente será copiar la `ssh key: "-----BEGIN RSA PRIVATE KEY-....--END RSA PRIVATE KEY-----"` y abrimos una terminal en nuestro equipo (no conectada a over the wire), creamos un archivo `nano bandit17_key`, pegamos la clave `ssh` y lo guardamos. Asignamos los permisos correctos al `nano: chmod 600 bandit17_key` y nos conectamos:

```
ssh -i bandit17_key bandit17@bandit.labs.overthewire.org -p 2220
```

Una vez dentro solo tendremos que buscar la contraseña con `cat /etc/bandit_pass/bandit17` y obtendremos la flag:

- `EReVavePLFHtFlFsjn3hyzMlvSuSAcRD`

NIVEL 17:



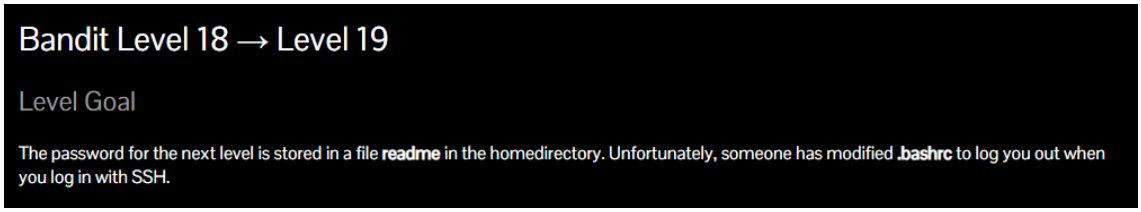
Comparamos de los archivos `passwords.new` y `passwords.old` con el comando `diff`, para ver los cambios de contraseña:

```
diff passwords.new passwords.old
```

Nos imprimirá las contraseñas que han sido cambiadas, (la nueva es la flag):

- `x2gLTTjFwMOhQ8oWNbMN362QKxfRqGlO`

NIVEL 18:



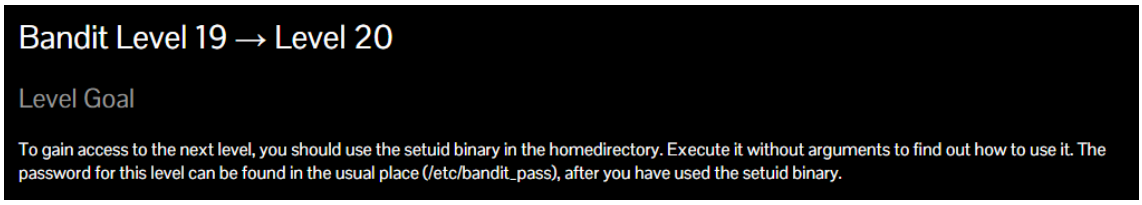
Al acceder al nivel, este nos echa. Para poder conseguir la flag de este nivel, debemos forzar la detención de la terminal y conseguir la flag antes de que nos eche. Introducimos un comando extra al conectarnos:

```
ssh bandit18@bandit.labs.overthewire.org -p 2220 cat readme
```

Nos saldrá de nuevo que introduzcamos la contraseña: `x2gLTTjFwMOhQ8oWNbMN362QKxfRqGlO`, y nos dará la flag para el siguiente nivel:

- `cGWpMaKXVwDUNgPAVJbWYuGHVn9zl3j8`

NIVEL 19:



Vemos que hay un archivo `bandit20-do`, si aplicamos el comando `ls-la` vemos que el propietario de ese archivo es `bandit20` y el grupo es `bandit19`, con las siglas `-rw-x-` significa que el grupo puede ser ejecutado, pero bajo el usuario `bandit20`, al ejecutarlo, `./bandit20-do` nos dice `run a command as another user` por lo tanto debemos buscar el archivo de contraseñas y abrir el usuario aplicando el comando:

```
./bandit20-do cat /etc/bandit_pass/bandit20
```

y nos dará la flag:

- `0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO`

NIVEL 20:

Bandit Level 20 → Level 21

Level Goal

There is a setuid binary in the homedirectory that does the following: it makes a connection to localhost on the port you specify as a commandline argument. It then reads a line of text from the connection and compares it to the password in the previous level (bandit20). If the password is correct, it will transmit the password for the next level (bandit21).

NOTE: Try connecting to your own network daemon to see if it works as you think

Creamos un servidor `nc` y nos conectamos con:

```
echo -n "0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO" | nc -l -p 12345
```

Mientras estamos en escucha abrimos otra terminal (tambien dentro de bandit) y nos conectaremos al servidor que hemos creado `./suconnect 12345` y lograremos interceptar la flag del siguiente nivel:

- `EeoULMCra2q0dSkYj561DX7s1CpBuOBt`

NIVEL 21:

Bandit Level 21 → Level 22

Level Goal

A program is running automatically at regular intervals from **cron**, the time-based job scheduler. Look in **/etc/cron.d/** for the configuration and see what command is being executed.

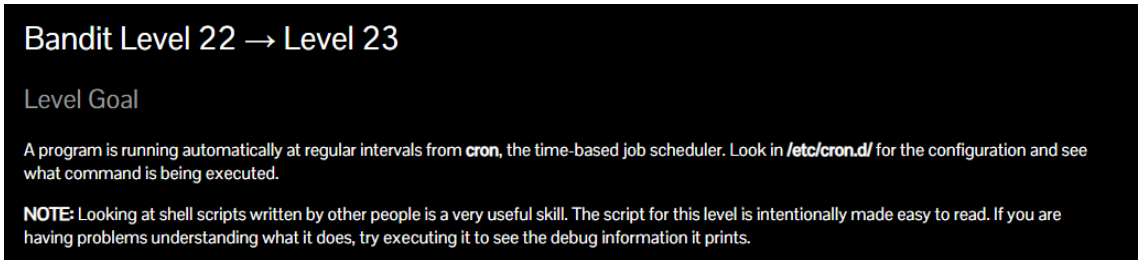
Tenemos que encontrar un script en la ruta `/etc/cron.d/`, aquí encontramos varios archivos cron, abrimos `cat cronjob_bandit22` y nos imprimirá un cron `* * * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null` aquí copiamos la ruta del `.sh` y lo abrimos `cat /usr/bin/cronjob_bandit22.sh` nos dará la ruta que contiene la flag:

```
cat /tmp/t7O61ds9S0RqQh9aMcz6ShpAoZKF7fgv
```

Nos dará la flag:

- `t7O61ds9S0RqQh9aMcz6ShpAoZKF7fg`

NIVEL 22:



Vamos al directorio cron y leemos el archivo `cat /etc/cron.d/cronjob_bandit23`, luego ejecutamos el comando `cat /usr/bin/cronjob_bandit23.sh` nos imprimirá un texto de cron con variables, aquí deberemos poner el comando que nos dice, pero con la variable de `$myname` como `bandit23` en este caso:

```
echo I am user bandit23 | md5sum | cut -d ' ' -f 1
```

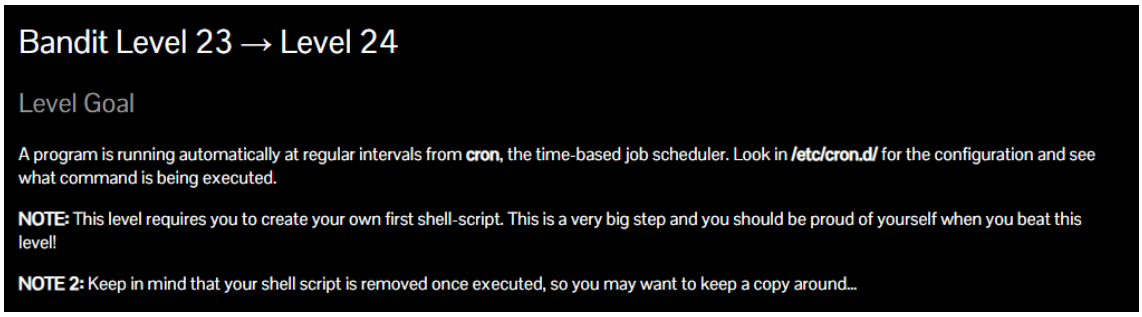
Nos imprimirá una serie de caracteres que deberemos leer en una ruta tmp:

```
cat /tmp/8ca319486bfbbc3663ea0fbe81326349
```

Nos dará la flag:

- `0Zf11ioIjMVN551jX3CmStKLYqjk54Ga`

NIVEL 23:



Cuando nos metemos en `cat /etc/cron.d/cronjob_bandit24` vemos que hay un `@reboot` en `bandit24`, si abrimos `cat /usr/bin/cronjob_bandit24.sh` vemos un script que nos dice que se está ejecutando y eliminando la flag en la ruta `/var/spool/$myname/foo`, cuando nos metemos, no podemos hacer nada porque no somos root, para poder vulnerar esto, nos dirigimos a la carpeta `/foo` con `cd /var/sppol/bandit24/foo` cuando hacemos `ls` para ver lo que hay en su interior, nos dice: `permission denied`. Debemos crear un archivo temporal y acceder desde ahí sin que actúe el script que elimina procesos. Creamos un script que nos permite interceptar la contraseña durante un tiempo y meterla

```
echo "cat /etc/bandit_pass/bandit24 > /tmp/pass24.txt" > grab.sh
```

en un archivo:

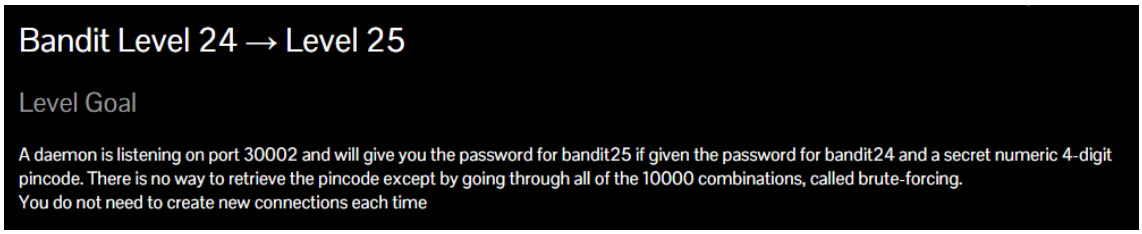
Seguidamente le otorgamos permisos para poder acceder:

```
chmod 777 grab.sh
```

accedemos al archivo `cat /tmp/pass24.txt` con esto debería aparecer la flag:

- `gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8`

NIVEL 24:



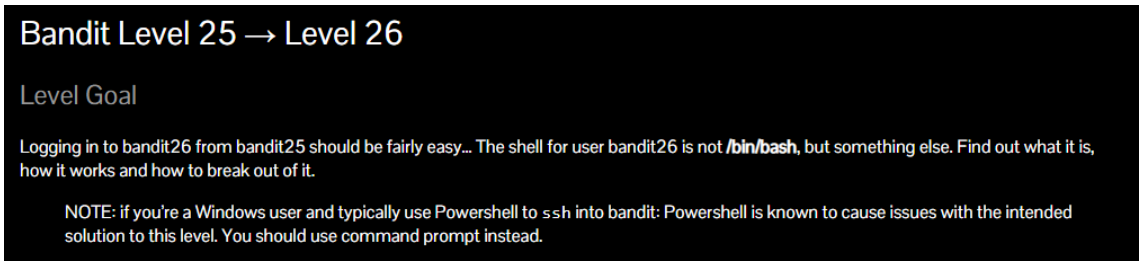
Tenemos una escucha en el puerto `30002` que nos dará la flag de `bandit25` si recibe la flag anterior + un código pin de 4 dígitos. La forma de adivinar el código es revisando las `10000` combinaciones (fuerza bruta), debemos conectarnos con el puerto, y ver que es lo que nos dice. Cuando introducimos algo erróneo aparece un mensaje: `Wrong!...`. Ahora que sabemos que nos responde, podemos empezar a desarrollar un script para intentar forzar el ping. Creamos una carpeta temporal `mktemp -d`, `cd /tmp/tmp.exsiIq05lU` , hacemos `export TERM=xterm`, seguidamente creamos `nano brute_force_ping.sh` y dentro ponemos el script:

```
#!/bin/bash
for i in {0000..9999}
do
        echo gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8 $i >> possibilities.txt
done
cat possibilities.txt | nc localhost 30002 > result.txt
```

Le damos permisos de ejecución `chmod +x brute_force_ping.sh` y lo ejecutamos `./brute_force_pin.sh`, nos dará un archivo `result.txt` que tendremos que filtrar con grep para hallar la flag: `sort result.txt | grep -v "Wrong!"` y nos dará la flag:

- `iCi86ttT4KSNe1armKiwbQNmB3YJP3q4`

NIVEL 25:



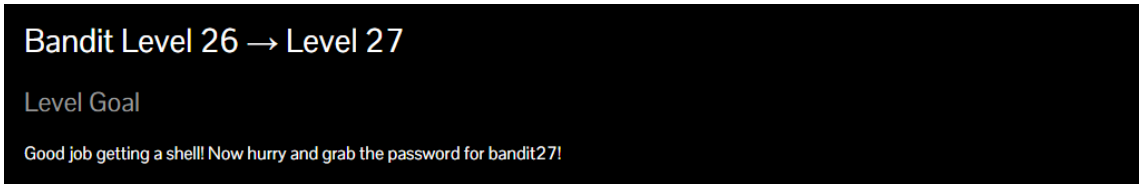
La terminal no está en `/bin/bash` tenemos que encontrar el script que nos diga en que formato se encuentra el texto, para ello, abrimos la carpeta `passwd` para ver su contenido filtrando por `bandit26`: `cat /etc/passwd | grep bandit26` aquí veremos un script el cual nos dice que la terminal está en `Linux`:

```
$ cat /usr/bin/showtext
#!/bin/sh

export TERM=linux

more ~/text.txt
exit 0
```

A continuación, intentaremos forzar la terminal `Linux` para que nos deje escribir comandos, para ello hacemos más pequeña la ventana para forzar `more%` y poder abrir en ese punto un editor `nano`. Ponemos: `ssh -i bandit26.sshkey -p 2220 bandit26@bandit.labs.overthewire.org` Rapidamente pulsamos la letra `v` forzando el editor de texto. ahí escribiremos: `:e cat /etc/bandit_pass/bandit26` y nos dará la flag:

- `s0773xxkk0MXfdqOfPRVr9L3jJBUOgCZ`

NIVEL 26:



En este nivel tenemos que volver a hacer pequeña la ventana para que nos aparezca el `more%` y presionamos la tecla `v` para forzar la escritura en la terminal. Ponemos el comando `:set shell=/bin/bash`, seguidamente ponemos `:shell` y nos desplegará la shell, hacemos comando `whoami` para saber bajo que nombre estamos entrando y nos pondrá `bandit26`, hacemos un `ls` para saber que archivos nos encontramos y vemos que hay uno que se llama `bandit27-do`, lo ejecutamos `./bandit27-do` y nos dirá `./bandit27-do id`. Por lo tanto solo debemos cambiar el ID por la ruta de la contraseña y le añadimos un `cat` para poder ver la flag; `./bandit27-do cat /etc/bandit_pass/bandit27` y nos dará la flag:

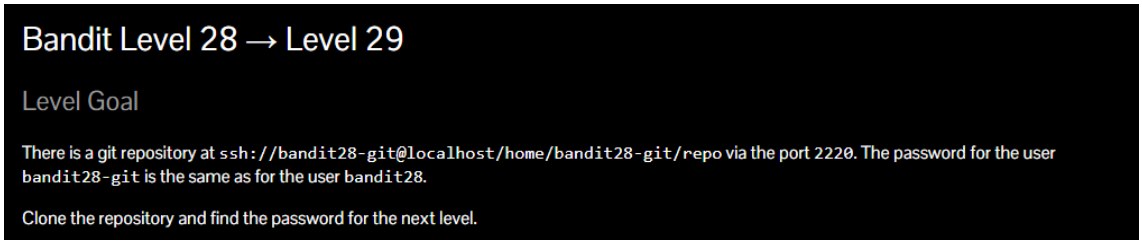
- `upsNCc7vzaRDx6oZC6GiR6ERwe1MowGB`

NIVEL 27:

Tenemos que clonar un repositorio `git` por el puerto `2220` y que su contraseña para clonarlo es la de `bandit27`. Lo primero que haremos será crear una carpeta temporal `mkdir -p /tmp/nivel27` y nos metemos en ella `cd /tmp/nivel27`, una vez dentro, clonamos el `git`: `git clone ssh://bandit27-git@localhost:2220/home/bandit27-git/repo`, una vez hecho y puesto la contraseña de `bandit27`, hacemos `ls` y vemos que hay un archivo `repo`, nos metemos con `cd` y abrimos el `README` que hay dentro, veremos la flag:

- `Yz9IpL0sBcCeuG7m9uQFt8ZNpS4HZRcN`

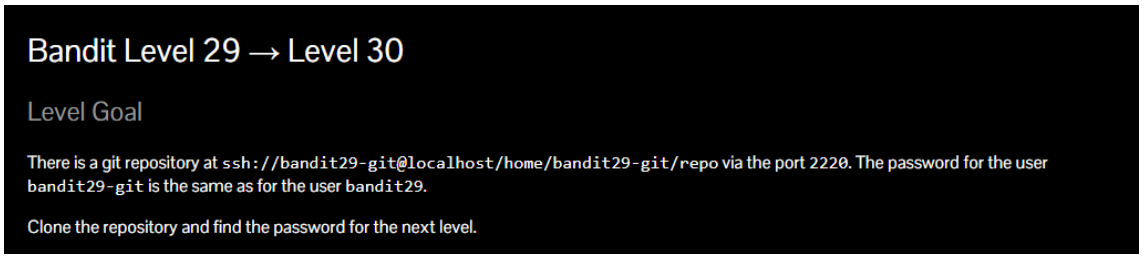
NIVEL 28:



Repetimos los mismos primeros pasos que el nivel anterior, creamos una carpeta temporal `mkdir -p /tmp/git2` y nos metemos en ella `cd /tmp/git2`, una vez dentro, clonamos el `git`: `git clone ssh://bandit28-git@localhost:2220/home/bandit28-git/repo` una vez hecho y puesto la contraseña anterior, hacemos `ls` y vemos que hay un archivo `repo`, nos metemos con `cd` y abrimos el `README.md` que hay dentro, aquí encontramos un usuario: `bandit29` pero la contraseña está oculta. Para descifrarla debemos ver si el archivo `git` se ha manipulado anteriormente, (con los llamados `Logs` (registros)), para ello nos metemos dentro del archivo `/repo` y ponemos el comando `git log`, vemos que la contraseña ha sido manipulada. Para ver mejor que es lo que ha cambiado, usamos `git log -p` que muestra la diferencia introducida en cada confirmación. También podemos usar `-i` que limita la salida a la última entrada. Si ejecutamos el comando `git log -p -i` veremos el cambio realizado y podemos encontrar la flag que había antes de ser manipulada:

- `4pT1t5DENaYuqnqvadYs1oE4QLCdjmJ7`

NIVEL 29:



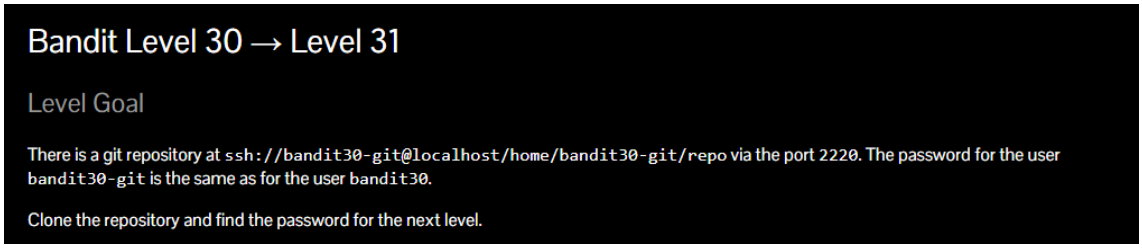
Este nivel es parecido a los anteriores, creamos la carpeta temporal `mkdir -p /tmp/git3` y nos metemos en ella `cd /tmp/git3`, una vez dentro, clonamos el `git`: `git clone ssh://bandit29-git@localhost:2220/home/bandit29-git/repo`, una vez hecho y puesto la contraseña anterior, hacemos `ls` y vemos que hay un archivo `repo`, nos metemos con `cd` y abrimos el `README.md` que hay dentro, donde encontramos un usuario: `bandit30` pero esta vez nos pone `<No password in production!>` en la contraseña, esto nos dice que hay varias ramas en este repositorio, para averiguar la flag. Debemos ver en cual se encuentra. Lo primero es saber en qué rama estamos poniendo el comando "`git branch`", en este caso estamos en `master*`, para saber cuántas ramas hay debemos poner el comando "`git branch -r`" donde nos aparecen:

```
origin/HEAD -> origin/master
  origin/dev
  origin/master
  origin/sploits-dev
```

, para cambiar de rama hacemos `git checkout dev`, cuando estemos en la rama `dev` hacemos `git log -p -i` para ver los registros de los commits anteriores y aquí podremos ver la flag:

- `qp30ex3VLz5MDG1n91YowTv4Q8l7CDZL`

NIVEL 30:



Creamos la carpeta temporal `mkdir -p /tmp/git30` y nos metemos en ella `cd /tmp/git30`, una vez dentro, clonamos el `git`: `git clone ssh://bandit30-git@localhost:2220/home/bandit30-git/repo` una vez hecho y puesto la contraseña anterior, hacemos `ls` y vemos que hay un archivo `repo`, nos metemos con `cd` y abrimos el `README.md` que hay dentro. Aquí encontramos un mensaje que dice: `just an epmty file... muahaha`. Vamos a probar con `git log -p`, vemos que hay una modificación, ahora vamos a ver las ramas con `git branch -r`, vamos a poner `git tag`, con esto lo que hacemos es ver si se ha etiquetado algún punto del repositorio. Vemos que nos aparece una palabra: `secret`, para ver que esconde la palabra ponemos `git show secret` y nos dará la flag:

- `fb5S2xb7bRyFmAvQYQGEqsbhVyJqhnDy`

NIVEL 31:

Creamos la carpeta temporal `mkdir -p /tmp/git31` y nos metemos en ella `cd /tmp/git31`, una vez dentro, clonamos el `git`: `git clone ssh://bandit31-git@localhost:2220/home/bandit31-git/repo` una vez hecho y puesto la contraseña anterior, hacemos `ls` y vemos que hay un archivo `repo`, nos metemos con `cd` y abrimos el `README.md` que hay dentro, aquí encontramos un mensaje que dice:

```
This time your task is to push a file to the remote repository.
Details:
    File name: key.txt
    Content: 'May I come in?'
  Branch: master
```

Tenemos que seguir estos pasos para subir este archivo, para ello debemos crear un archivo que contenga el mensaje: `touch key.txt` y seguidamente ponemos el comando `echo "Msy I come in?" > key.txt`, después añadimos `key.txt` al `git` aplicando `git add key.txt`, nos dice que se ha creado un archivo `.gitignore`, lo eliminamos `rm .gitignore` volvemos a hacer `git add key.txt` y seguidamente lo comentamos `git commit -m "subir archivo"` y subimos el archivo con `git push origin master` ponemos la contraseña del nivel 31 y nos dará la flag:

- `3O9RfhqyAlVBEZpVb6LYStshZoqoSx5K`

NIVEL 32:

La terminal nos convierte cada comando en letras mayúsculas, necesitamos arreglarlo para volver a tener una `shell` normal, mediante el comando `$0` hacemos que se arregle, hacemos `pwd` para saber dónde nos encontramos y ponemos `cat /etc/bandit_pass/bandit33` para que nos de la flag:

- `tQdtbs5D5i2vJwkO8mEyYEyTL8izoeJ0`

(para salir de la terminal `ctrl+C`).

BANDIT COMPLETADO!