Level 0 → 1

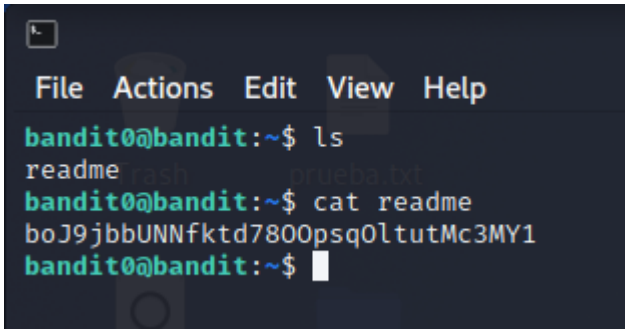
Level Goal

The password for the next level is stored in a file called **readme** located in the home directory. Use this password to log into bandit1 using SSH. Whenever you find a password for a level, use SSH (on port 2220) to log into that level and continue the game.

Commands you may need to solve this level

ls, cd, cat, file, du, find

First we connect with:
ssh bandit0@bandit.labs.overthewire.org -p 2220.
The rubric tell us that there is a password stored in a file called **readme**. We check what files are in our current directory with a `ls`, we can see a readme file. Then we open it with a `cat readme`



Level 1 → 2

username: bandit1
password: boJ9jbbUNNfktd78OOpsqOltutMc3MY1

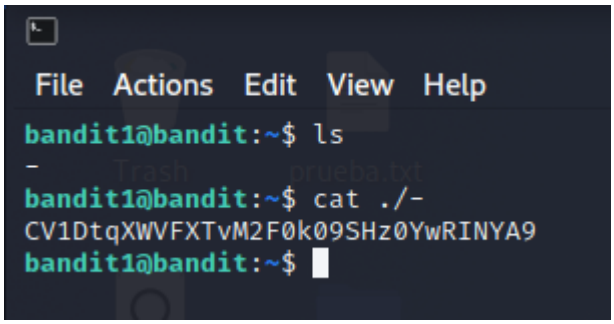
Level Goal

The password for the next level is stored in a file called - located in the home directory

Commands you may need to solve this level

ls, cd, cat, file, du, find

We can do the same as before, but at executing the `cat -` command we can see that opening files with special characters is different. In this case we can use the command with a ./"filenamewithspecialcharacters".



Level 2 → 3

username: bandit2
password: CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

Bandit Level 2 → Level 3

Level Goal

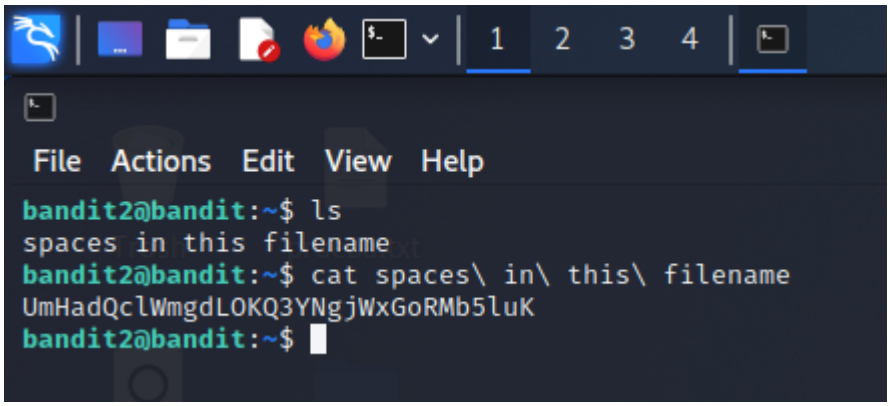
The password for the next level is stored in a file called **spaces in this filename** located in the home directory

Commands you may need to solve this level

ls, cd, cat, file, du, find

As level 2 we only have to do a cat, but again, having a file with spaces in his filename have a different way to execute it. In this case we can:

- Start writing: sp... and tab it.

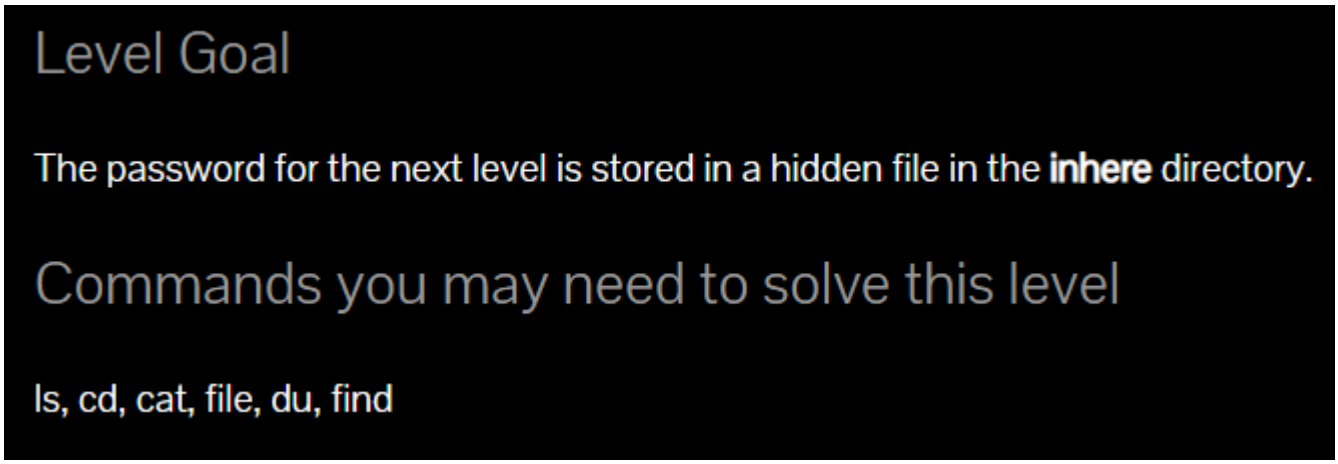


- Execute the cat with "spaces in this filename", this will open LITERALLY the "spaces in this filename"

```
bandit2@bandit:~$ cat "spaces in this filename"
UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK
bandit2@bandit:~$
```

Level 3→4

username: bandit2
password: UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK



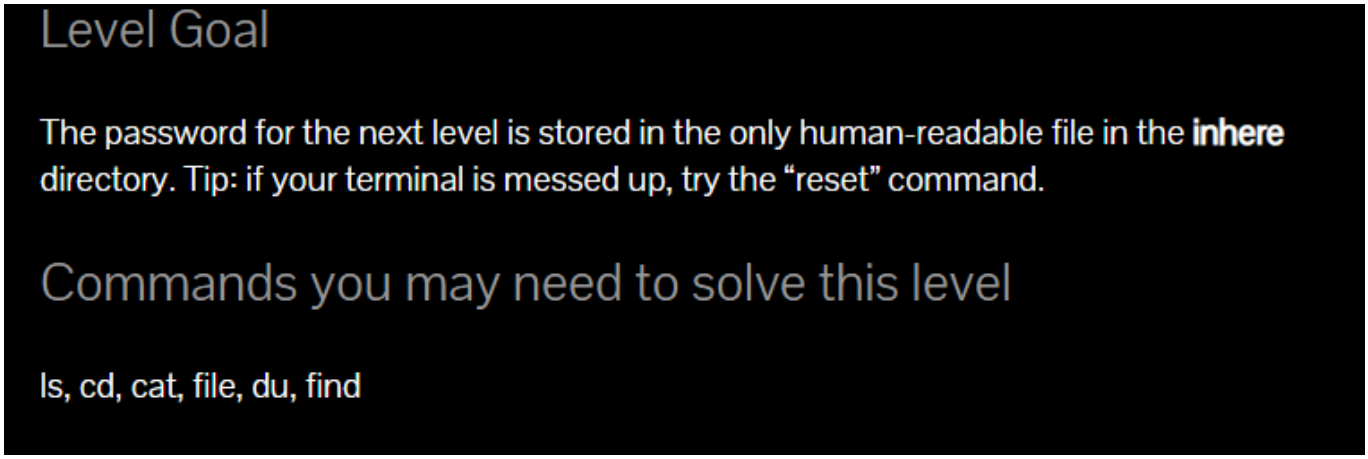
In this level we will learn how to move between directories using `cd --> Change Directory`.
First we ennumerate what is in the inhere directory and then we can do an ls to display all the files of inhere, but this time there are hidden files that we can see with `ls -al`. Finally we see a .hidden file, with all this information que can

execute a cat.

```
bandit3@bandit:~$ ls -al inhere/
total 12
drwxr-xr-x 2 root    root    4096 May  7  2020 .
drwxr-xr-x 3 root    root    4096 May  7  2020 ..
-rw-r----- 1 bandit4 bandit3   33 May  7  2020 .hidden
bandit3@bandit:~$ cat inhere/.hidden
pIwrPrtPN36QITSp3EQaw936yaFoFgAB
bandit3@bandit:~$
```

Level 4 → 5

username: bandit4
password: pIwrPrtPN36QITSp3EQaw936yaFoFgAB



In this level they give us the tip: "only human-readable" this is very ambiguous, but we can try to do a file of every file in the inhere directory to see if the directories are in "ASCII text". Maybe you can try with the `find` command, but there we can see if the file has **read** permissions, not if the content is **readable**.

```
bandit4@bandit:~$ file inhere/*
inhere/-file00: data
inhere/-file01: data
inhere/-file02: data
inhere/-file03: data
inhere/-file04: data
inhere/-file05: data
inhere/-file06: data
inhere/-file07: ASCII text
inhere/-file08: data
inhere/-file09: data
bandit4@bandit:~$ cat inhere/-file07
koReBOKuIDDepwhWk7jZC0RTdopnAYKh
bandit4@bandit:~$
```

We can try with the `strings` command. Strings command will have as an output all the "human-readable" characters of every file.

```
bandit4@bandit:~$ strings inhere/*
!TQO
koReBOKuIDDepwhWk7jZC0RTdopnAYKh
bandit4@bandit:~$
```

It worked aswell.

Level 5 → 6

username: bandit5
password: koReBOKuIDDepwhWk7jZC0RTdopnAYKh*

Level Goal

The password for the next level is stored in a file somewhere under the **inhere** directory and has all of the following properties:

human-readable
1033 bytes in size
not executable

Commands you may need to solve this level

ls, cd, cat, file, du, find

We can see that "inhere" there are TONS of files and directories, so we must use the `find` command. We can put the flag -size 1033c , c represents bytes. For the not executable we can use the flag -executable, but as we are looking for a NON executable file we must put a ! before. and for the readable condition we will use the flag -readable. Putting everything together it will give us the file where it is stored the flag, then we will open it.

```
bandit5@bandit:~$ find inhere/ -size 1033c ! -executable -readable
inhere/maybehere07/.file2
bandit5@bandit:~$ cat inhere/maybehere07/.file2
DXjZPULLxYr17uwoI01bNLQbtFemEgo7
```

Level 6 → 7

username: bandit6
password: DXjZPULLxYr17uwoI01bNLQbtFemEgo7

Level Goal

The password for the next level is stored **somewhere on the server** and has all of the following properties:

owned by user bandit7
owned by group bandit6
33 bytes in size

Commands you may need to solve this level

ls, cd, cat, file, du, find, grep

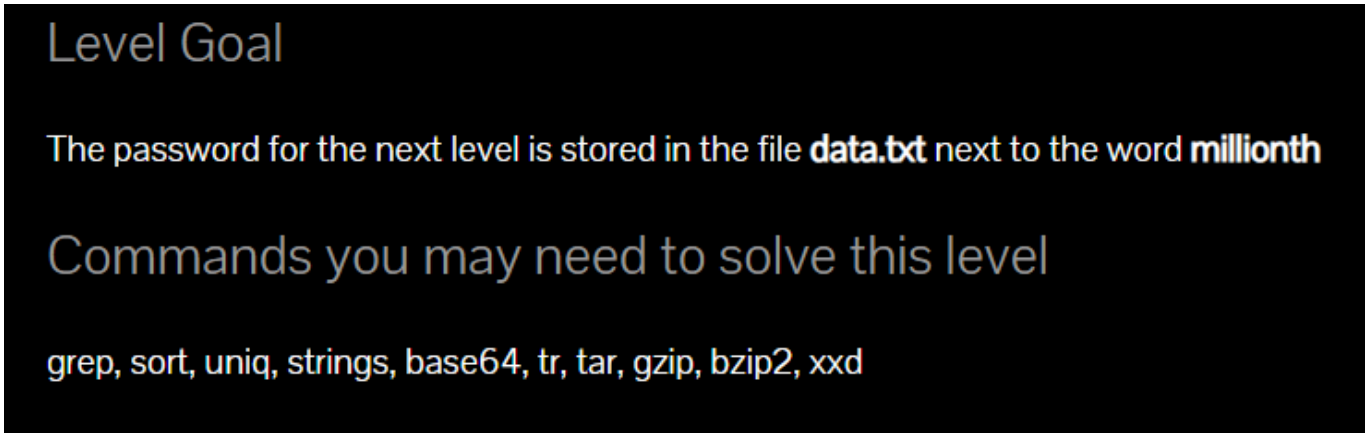
This level has the same methodology as level 5, we will execute a `find` command. Now we will use **size** again, looking for 33 bytes (find -size 33c). And the new two flags we will be introducing now:

- -group → looks for the group
- -user → looks for the user.
  Now we must use them in the / directory, because the rubrics says that it is stored **somewhere on the server** , so executing the find on root will be the most intelligent way right here. We will se there are a lot of files with the message "Permission denied", for ignoring them we can use a 2>/dev/null, that sends all the error messages to /dev/null and we will only see the other messages.

```
bandit6@bandit:~$ find / -size 33c -group bandit6 -user bandit7  2>/dev/null
/var/lib/dpkg/info/bandit7.password
bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password
HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs
bandit6@bandit:~$
```

Level 7 → 8

username: bandit7
password: HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs

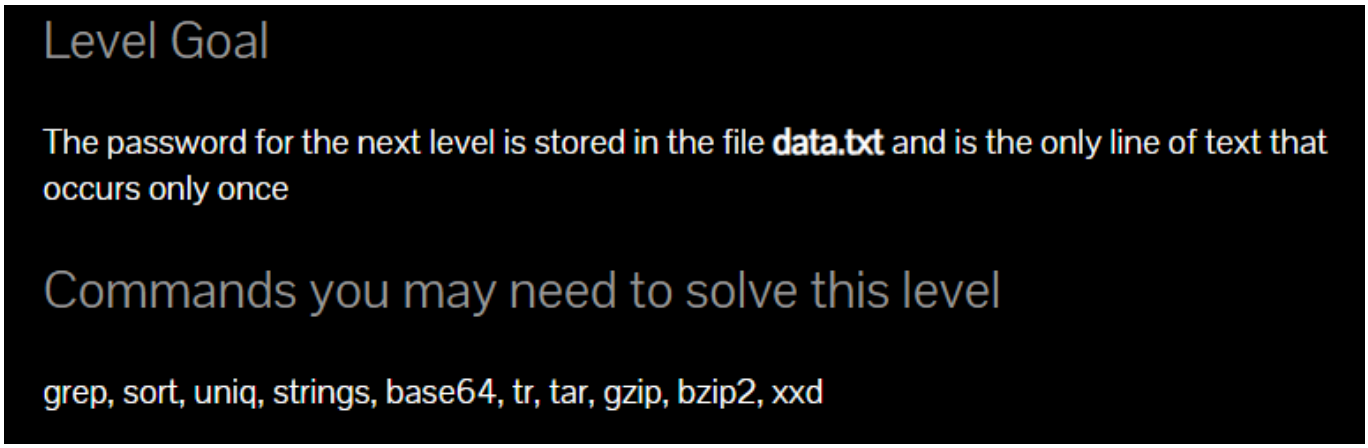


For this level we will use the grep command, that looks for patterns, now we will look for the word "millionth".

```
bandit7@bandit:~$ grep "millionth" data.txt
millionth       cvX2JJa4CFALtqS87jk27qwqGhBM9plV
bandit7@bandit:~$
```

Level 8 → 9

username: bandit8
password: cvX2JJa4CFALtqS87jk27qwqGhBM9plV

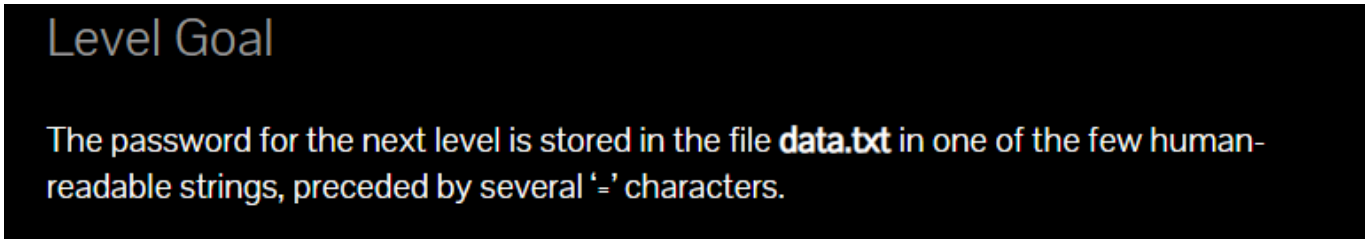


For this level first we are going to use the `uniq` command, it give us the lines that occur only once, but first we have to `sort` it for making it work. With uniq -u we will print only the lines that occur once.

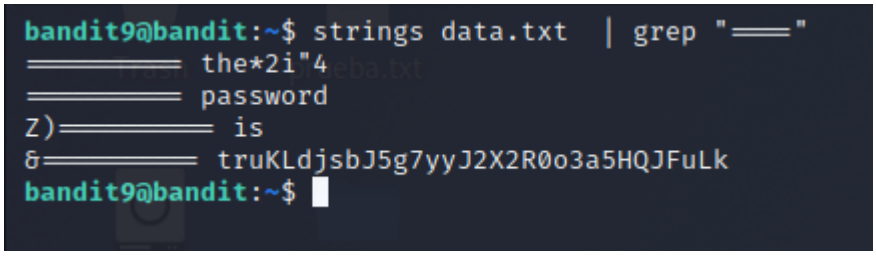
```
bandit8@bandit:~$ sort data.txt | uniq -u
UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR
bandit8@bandit:~$
```

Level 9 → 10

username: bandit9
password: UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR

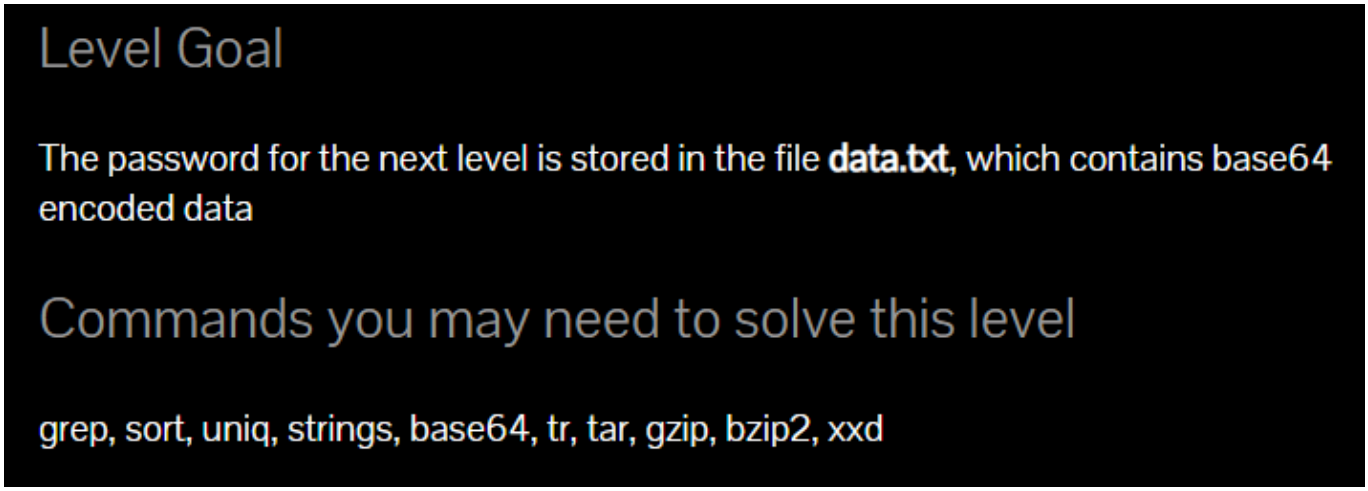


As we have seen before we can use the command `strings` to see the human-readable strings, and with grep we will look for a pattern with a few "==== " before.

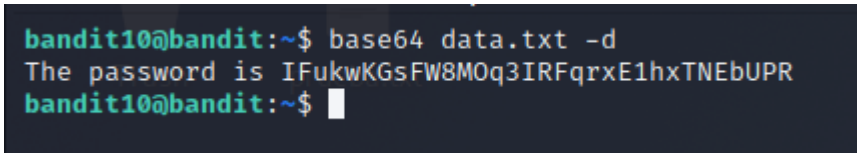


Level 10 → 11

username: bandit10
password: truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk



We have the command `base64` with the flag `-d` that decrypts. If we execute it we will have the password.



Level 11 → 12

username: bandit11
password: IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR

Level Goal

The password for the next level is stored in the file **data.txt**, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions

Commands you may need to solve this level

grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd

with the command`tr` we change one thing for another. In this case we want to to a typical 'rot13'.
ABCDEFGHIJKLMNÑOPQRSTUVWXYZ
We want to change A for N
and for the lowercase case
abcdefghijklmnñopqrstuvwxyz
a for n

The command will finally be

```
tr 'A-Za-z' 'N-ZA-Mn-za-m'
```

```
bandit11@bandit:~$ cat data.txt| tr 'A-Za-z' 'N-ZA-Mn-za-m'
The password is 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu
bandit11@bandit:~$
```

Level 12 → 13

username: bandit12
password: 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu

Level Goal

The password for the next level is stored in the file **data.txt**, which is a hexdump of a file that has been repeatedly compressed. For this level it may be useful to create a directory under /tmp in which you can work using mkdir. For example: mkdir /tmp/myname123. Then copy the datafile using cp, and rename it using mv (read the manpages!)

Commands you may need to solve this level

grep, sort, uniq, strings, base64, tr, tar, gzip, bzip2, xxd, mkdir, cp, mv, file

There are 2 ways of doing this level but concept is the same.
First we will create a mktemp directory to operate. then we are going to copy the file to our directory.

```
bandit12@bandit:~$ mktemp -d
/tmp/tmp.3mPbkDeAN2
bandit12@bandit:~$ cd /tmp/tmp.3mPbkDeAN2
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$ mv data.txt data.hex
mv: cannot stat 'data.txt': No such file or directory
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$ cp /home/bandit12/data.txt /tmp/tmp.3mPbkDeAN2
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$ ls
data.txt
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$ mv data.txt data.hex
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$
```

We have to first reverse the hexdump with the command `xxd -d "file"` , for doing this we have to change the name to a .hex and then execute the comand. Creating a new file.

```
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$ xxd -r data.hex data
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$ ls
data  data.hex
bandit12@bandit:/tmp/tmp.3mPbkDeAN2$
```

From this point we can do it manually, checking everytime what type of file we have to decompress or we can make an script. That does it automatically.
We can copy the file to our local machine and use other tools like 7z, then we will not have to use gzip, bzip2, tar... That is what i'm going to do
After copying it im going to do a `7z l "file"` for knowing if I decompress it what file im going to get.

```
┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# 7z l data.gzip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 606 bytes (1 KiB)

Listing archive: data.gzip

--
Path = data.gzip
Type = gzip
Headers Size = 20

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2020-05-07 14:14:30 .....          573          606  data2.bin
------------------- ----- ------------ ------------  ------------------------
2020-05-07 14:14:30                573          606  1 files
```

Now is when we can do it manually or making a script.

Manually:

```
┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# 7z x data.gzip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 606 bytes (1 KiB)

Extracting archive: data.gzip
--
Path = data.gzip
Type = gzip
Headers Size = 20

Everything is Ok

Size:       573
Compressed: 606

┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# ls
data2.bin  data.gzip  data.hex  decompressor.sh  decompressor.sh.save
```

we decompress untill we see it cannot be decompressed anymore

```
┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# ls
data2  data2.bin  data.gzip  data.hex  decompressor.sh  decompressor.sh.save

┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# mv data2 data2.gzip

┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# 7z l data2.gzip

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 431 bytes (1 KiB)

Listing archive: data2.gzip

--
Path = data2.gzip
Type = gzip
Headers Size = 20

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2020-05-07 14:14:30 .....        20480          431  data4.bin
------------------- ----- ------------ ------------  ------------------------
2020-05-07 14:14:30               20480          431  1 files
```

We got a file that we don't know what it is, we execute `file`, and we change the name and keep going.

```
┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# file data9.bin
data9.bin: ASCII text

┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# 7z l data9.bin

7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz (706E5),ASM,AES-NI)

Scanning the drive for archives:
1 file, 49 bytes (1 KiB)

Listing archive: data9.bin


ERROR: data9.bin : Can not open the file as archive



Errors: 1
```

We finally get into a file that is ASCII text, and can't be decompressed anymore. This might be the flag.

```
┌──(root💀kali)-[/home/kali/Desktop/scripts]
└─# cat data9.bin
The password is 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL
```

Level 13 → 14

username: bandit13
password: 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL

Level Goal

The password for the next level is stored in **/etc/bandit_pass/bandit14 and can only be read by user bandit14**. For this level, you don't get the next password, but you get a private SSH key that can be used to log into the next level. **Note: localhost** is a hostname that refers to the machine you are working on

Commands you may need to solve this level

ssh, telnet, nc, openssl, s_client, nmap

Now we are going to use SSH keys. First we can see that this is a private key.

```
bandit13@bandit:~$ ls -al sshkey.private
-rw-r----- 1 bandit14 bandit13 1679 May  7  2020 sshkey.private
```

it seems like the owner is bandit14, so we can try to connect through ssh with the following command.

`ssh -i "privatekey" user@hostname0

```
bandit13@bandit:~$ ssh -i sshkey.private bandit14@localhost
Could not create directory '/home/bandit13/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit13/.ssh/known_hosts).
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

Linux bandit.otw.local 5.4.8 x86_64 GNU/Linux

Welcome to OverTheWire!
```

Now that we are bandit14 we can get the password from /etc/bandit_pass/bandit14

```
bandit14@bandit:~$ cat /etc/bandit_pass/bandit14
4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e
bandit14@bandit:~$
```

Level 14 → 15

username: bandit15
password: 4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e

Level Goal

The password for the next level can be retrieved by submitting the password of the current level to **port 30000 on localhost**.

Commands you may need to solve this level

ssh, telnet, nc, openssl, s_client, nmap

In this level we will use nc (netcut) to send a message to a port.

```
bandit14@bandit:~$ echo "4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e" | nc localhost 30000
Correct!
BfMYroe26WYalil77FoDi9qh59eK5xNr
```

Level 15 → 16

username: bandit15
password: BfMYroe26WYalil77FoDi9qh59eK5xNr

The password for the next level can be retrieved by submitting the password of the current level to **port 30001 on localhost** using SSL encryption.

Helpful note: Getting "HEARTBEATING" and "Read R BLOCK"? Use -ign_eof and read the "CONNECTED COMMANDS" section in the manpage. Next to 'R' and 'Q', the 'B' command also works in this version of that command...

Commands you may need to solve this level

ssh, telnet, nc, openssl, s_client, nmap

We can't submit the pass with an `echo "password" | nc name port` because it is encrypted with ssl, so we will have to connect through ssl with openssl and s_client to the port said before.

```
bandit15@bandit:~$ openssl s_client -connect localhost:30001
CONNECTED(00000003)
depth=0 CN = localhost
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = localhost
verify return:1
---
Certificate chain
 0 s:/CN=localhost
   i:/CN=localhost
```

```
BfMYroe26WYalil77FoDi9qh59eK5xNr
Correct!
cluFn7wTiGryunymYOu4RcffSxQluehd
```

Then we submitt the password.

Level 16 → 17

username: bandit16
password: cluFn7wTiGryunymYOu4RcffSxQluehd

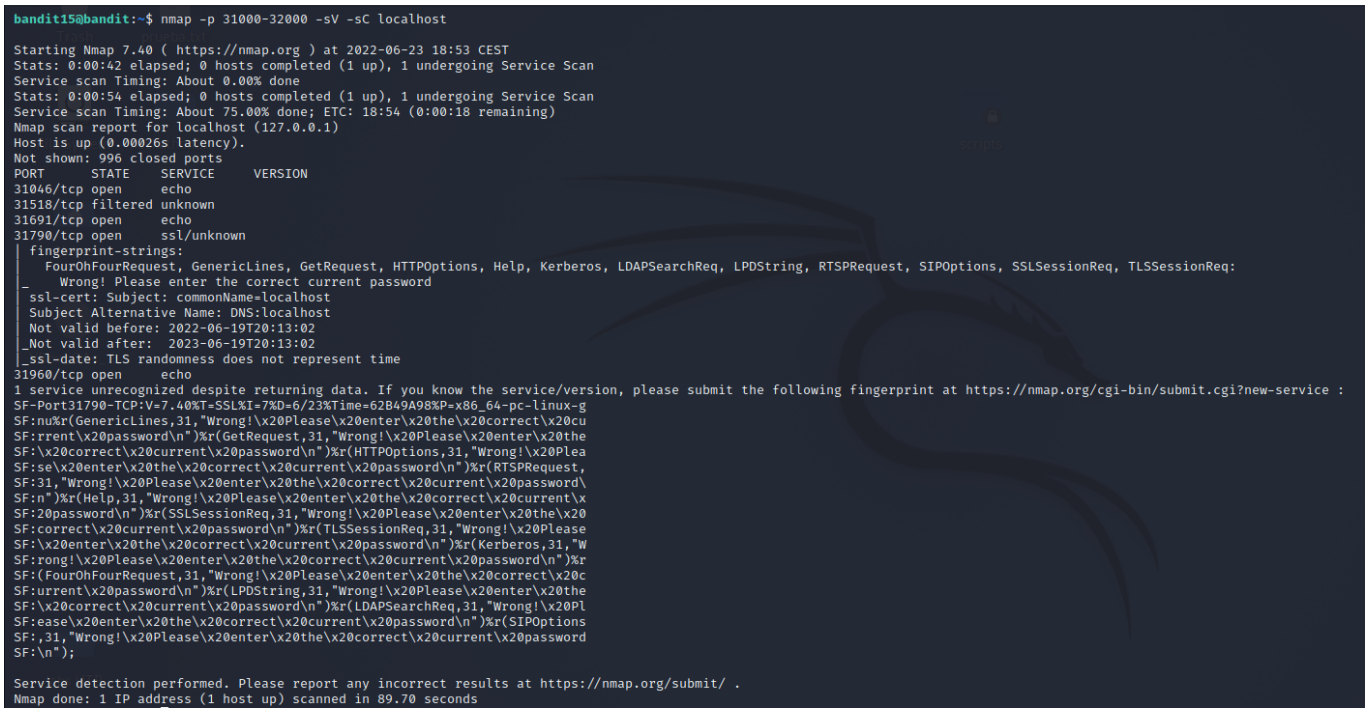
Level Goal

The credentials for the next level can be retrieved by submitting the password of the current level to **a port on localhost in the range 31000 to 32000**. First find out which of these ports have a server listening on them. Then find out which of those speak SSL and which don't. There is only 1 server that will give the next credentials, the others will simply send back to you whatever you send to it.

Commands you may need to solve this level

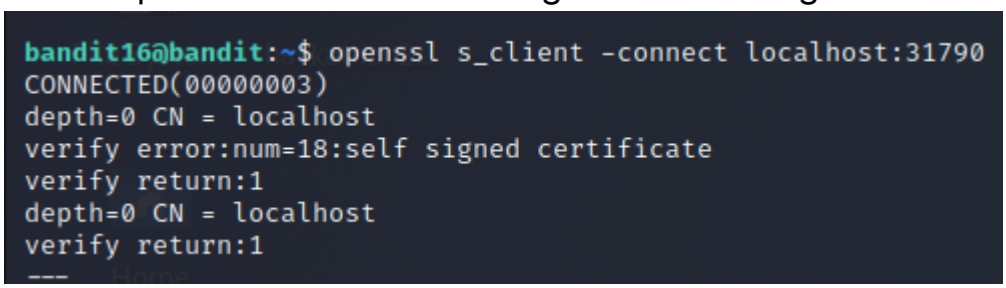
ssh, telnet, nc, openssl, s_client, nmap

We are going to do an nmap to know what ports are open.



```
bandit15@bandit:~$ nmap -p 31000-32000 -sV -sC localhost

Starting Nmap 7.40 ( https://nmap.org ) at 2022-06-23 18:53 CEST
Stats: 0:00:42 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Stats: 0:00:54 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 75.00% done; ETC: 18:54 (0:00:18 remaining)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00026s latency).
Not shown: 996 closed ports
PORT      STATE    SERVICE     VERSION
31046/tcp open     echo
31518/tcp filtered unknown
31691/tcp open     echo
31790/tcp open     ssl/unknown
| fingerprint-strings:
|   FourOhFourRequest, GenericLines, GetRequest, HTTPOptions, Help, Kerberos, LDAPSearchReq, LPDString, RTSPRequest, SIPOptions, SSLSessionReq, TLSSessionReq:
|_    Wrong! Please enter the correct current password
| ssl-cert: Subject: commonName=localhost
| Subject Alternative Name: DNS:localhost
| Not valid before: 2022-06-19T20:13:02
|_Not valid after:  2023-06-19T20:13:02
|_ssl-date: TLS randomness does not represent time
31960/tcp open     echo
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-Port31790-TCP:V=7.40%T=SSL%I=7%D=6/23%Time=62B49A98%P=x86_64-pc-linux-g
SF:nu%r(GenericLines,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20cu
SF:rrent\x20password\n")%r(GetRequest,31,"Wrong!\x20Please\x20enter\x20the
SF:\x20correct\x20current\x20password\n")%r(HTTPOptions,31,"Wrong!\x20Plea
SF:se\x20enter\x20the\x20correct\x20current\x20password\n")%r(RTSPRequest,
SF:31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\
SF:n")%r(Help,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x
SF:20password\n")%r(SSLSessionReq,31,"Wrong!\x20Please\x20enter\x20the\x20
SF:correct\x20current\x20password\n")%r(TLSSessionReq,31,"Wrong!\x20Please
SF:\x20enter\x20the\x20correct\x20current\x20password\n")%r(Kerberos,31,"W
SF:rong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\n")%r
SF:(FourOhFourRequest,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20c
SF:urrent\x20password\n")%r(LPDString,31,"Wrong!\x20Please\x20enter\x20the
SF:\x20correct\x20current\x20password\n")%r(LDAPSearchReq,31,"Wrong!\x20Pl
SF:ease\x20enter\x20the\x20correct\x20current\x20password\n")%r(SIPOptions
SF:,31,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x20password
SF:\n");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 89.70 seconds
```

We can see that port 31790 has a ssl service, we are going to send him our current password and see if he gives us the flag.



```
bandit16@bandit:~$ openssl s_client -connect localhost:31790
CONNECTED(00000003)
depth=0 CN = localhost
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = localhost
verify return:1
---
```

We can connect but it gives us a private key

```
-----BEGIN RSA PRIVATE KEY-----
MIIEogIBAAKCAQEAvmOkuifmMg6HL2YPIOjon6iWfbp7c3jx34YkYWqUH57SUdyJ
imZzeyGC0gtZPGujUSxiJSWI/oTqexh+cAMTSMlOJf7+BrJObArnxd9Y7YT2bRPQ
Ja6Lzb558YW3FZl87ORiO+rW4LCDCNd2lUvLE/GL2GWyuKN0K5iCd5TbtJzEkQTu
DSt2mcNn4rhAL+JFr56o4T6z8WWAW18BR6yGrMq7Q/kALHYW3OekePQAzL0VUYbW
JGTi65CxbCnzc/w4+mqQyvmzpWtMAzJTzAzQxNbkR2MBGySxDLrjg0LWN6sK7wNX
x0YVztz/zbIkPjfkU1jHS+9EbVNj+D1XFOJuaQIDAQABAoIBABagpxpM1aoLWfvD
KHcj10nqcoBc4oE11aFYQwik7xfW+24pRNuDE6SFthOar69jp5RlLwD1NhPx3iBl
J9nOM8OJ0VToum43UOS8YxF8WwhXriYGnc1sskbwpXOUDc9uX4+UESzH22P29ovd
d8WErY0gPxun8pbJLmxkAtWNhpMvfe0050vk9TL5wqbu9AlbssgTcCXkMQnPw9nC
YNN6DDP2lbcBrvgT9YCNL6C+ZKufD52yOQ9qOkwFTEQpjtF4uNtJom+asvlpmS8A
vLY9r60wYSvmZhNqBUrj7lyCtXMIu1kkd4w7F77k+DjHoAXyxcUp1DGL51sOmama
+TOWWgECgYEA8JtPxP0GRJ+IQkX262jM3dEIkza8ky5moIwUqYdsx0NxHgRRhORT
8c8hAuRBb2G82so8vUHk/fur85OEfc9TncnCY2crpoqsghifKLxrLgtT+qDpfZnx
SatLdt8GfQ85yA7hnWWJ2MxF3NaeSDm75Lsm+tBbAiyc9P2jGRNtMSkCgYEAypHd
HCctNi/FwjulhttFx/rHYKhLidZDFYeiE/v45bN4yFm8x7R/b0iE7KaszX+Exdvt
SghaTdcG0Knyw1bpJVyusavPzpaJMjdJ6tcFhVAbAjm7enCIvGCSx+X3l5SiWg0A
R57hJglezIiVjv3aGwHwvlZvtszK6zV6oXFAu0ECgYAbjo46T4hyP5tJi93V5HDi
Ttiek7xRVxUl+iU7rWkGAXFpMLFteQEsRr7PJ/lemmEY5eTDAFMLy9FL2m9oQWCg
R8VdwSk8r9FGLS+9aKcV5PI/WEKlwgXinB3OhYimtiG2Cg5JCqIZFHxD6MjEGOiu
L8ktHMPvodBwNsSBULpG0QKBgBAplTfC1HOnWiMGOU3KPwYWt0O6CdTkmJOmL8Ni
blh9elyZ9FsGxsgtRBXRsqXuz7wtsQAgLHxbdLq/ZJQ7YfzOKU4ZxEnabvXnvWkU
YOdjHdSOoKvDQNWu6ucyLRAWFuISeXw9a/9p7ftpxm0TSgyvmfLF2MIAEwyzRqaM
77pBAoGAMmjmIJdjp+Ez8duyn3ieo36yrttF5NSsJLAbxFpdlc1gvtGCWW+9Cq0b
dxviW8+TFVEBl1O4f7HVm6EpTscdDxU+bCXWkfjuRb7Dy9GOtt9JPsX8MBTakzh3
vBgsyi/sN3RqRBcGU40fOoZyfAMT8s1m/uYv52O6IgeuZ/ujbjY=
-----END RSA PRIVATE KEY-----

closed
bandit16@bandit:~$
```

Now we copy this private key.

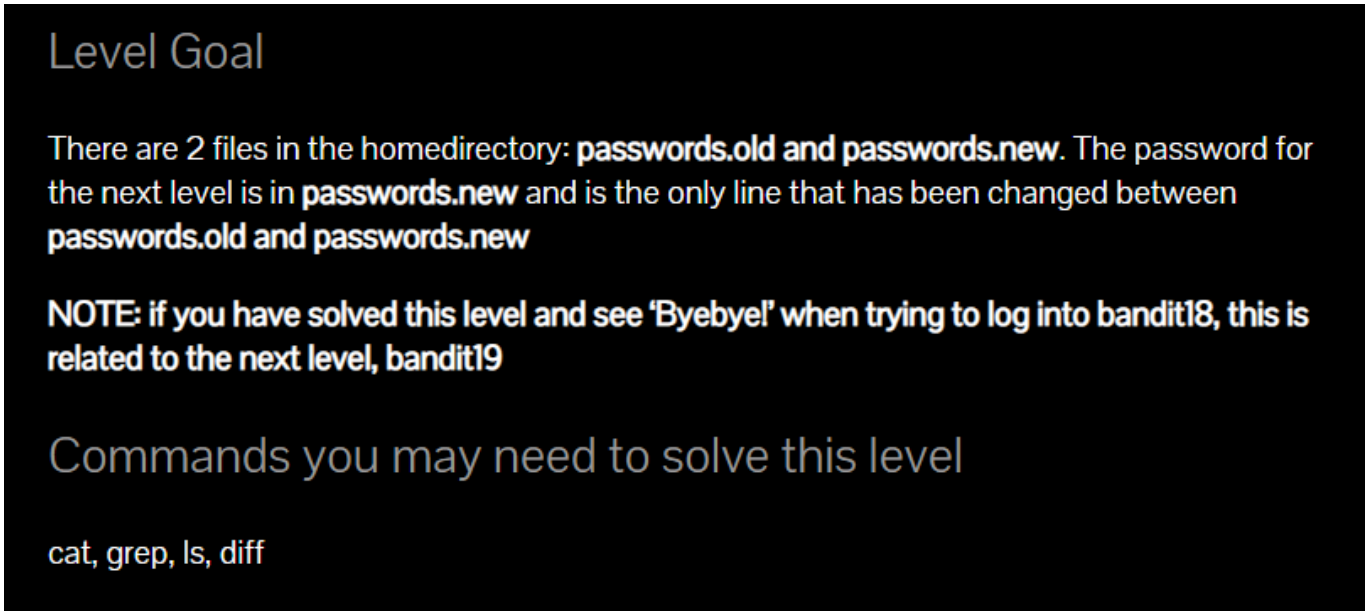
```
bandit16@bandit:~$ touch clave.private
touch: cannot touch 'clave.private': Permission denied
bandit16@bandit:~$ mktemp -d
/tmp/tmp.kyKgaqX8yM
bandit16@bandit:~$ cd /tmp/tmp.kyKgaqX8yM
bandit16@bandit:/tmp/tmp.kyKgaqX8yM$ touch clave.private
bandit16@bandit:/tmp/tmp.kyKgaqX8yM$ nano clave.private
Unable to create directory /home/bandit16/.nano: Permission denied
It is required for saving/loading search history or cursor positions.

Press Enter to continue

bandit16@bandit:/tmp/tmp.kyKgaqX8yM$ ssh -i clave.private bandit17@localhost
```

The key must have permissions 600, and then we can connect to bandit17 and have his password.

```
bandit17@bandit:~$ cat /etc/bandit_pass/bandit17
xLYVMN9WE5zQ5vHacb0sZEVqbrp7nBTn
bandit17@bandit:~$
```

Level 17 → 18

username: bandit17
password: xLYVMN9WE5zQ5vHacb0sZEVqbrp7nBTn

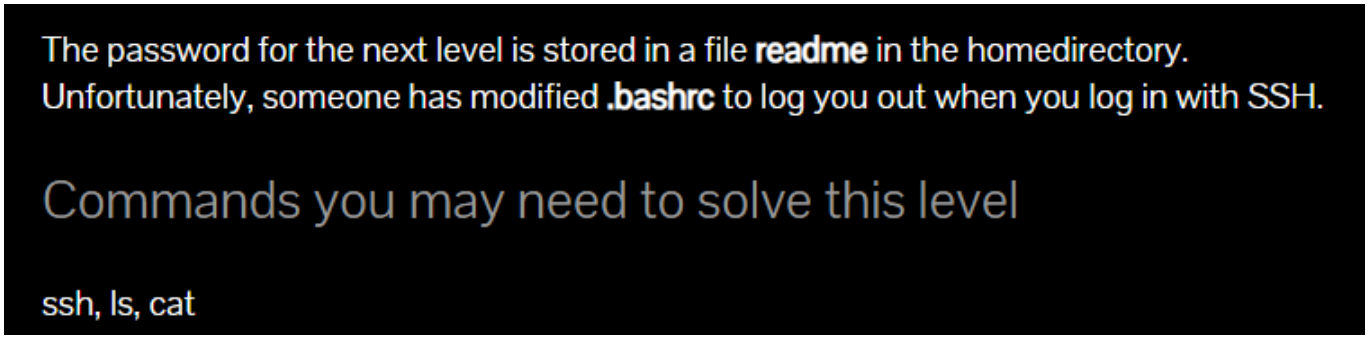



```
bandit17@bandit:~$ diff passwords.old passwords.new
42c42
< w0Yfolrc5bwjS4qw5mq1nnQi6mF03bii
---
> kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd
```

With the command diff que can see what differs from one file to another.

Level 18 → 19

username: bandit18
password: kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd



As we know **bash.rc** has been modified, but when we connect with ssh it has some delay before kicking us out, so in that time we can execute some

commands. We will try:

```
└─$ ssh bandit18@bandit.labs.overthewire.org -p 2220 cat "readme"
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit18@bandit.labs.overthewire.org's password:
IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x
```

Level 19 → 20

username: bandit19
password: IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x

Level Goal

To gain access to the next level, you should use the setuid binary in the homedirectory. Execute it without arguments to find out how to use it. The password for this level can be found in the usual place (/etc/bandit_pass), after you have used the setuid binary.

```
bandit19@bandit:~$ ls
bandit20-do
bandit19@bandit:~$ ls -al bandit20-do
-rwsr-x--- 1 bandit20 bandit19 7296 May  7  2020 bandit20-do
bandit19@bandit:~$ file bandit20-do
bandit20-do: setuid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=8e941f24b8c5cd0af67b22b724c57e1ab92a92a1, not stripped
bandit19@bandit:~$
```

We can see that this is a file with SUID characteristics, as group bandit19 we can execute it. This binary allows us to execute commands being bandit20.

```
bandit19@bandit:~$ ./bandit20-do cat "/etc/bandit_pass/bandit20"
GbKksEFF4yrVs6il55v6gwY5aVje5f0j
```

Level 20 → 21

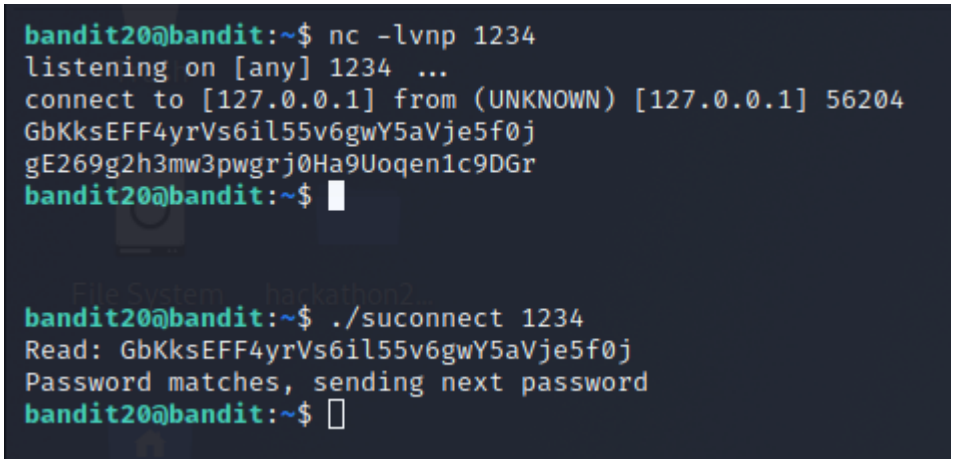
username: bandit20
password: GbKksEFF4yrVs6il55v6gwY5aVje5f0j

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).

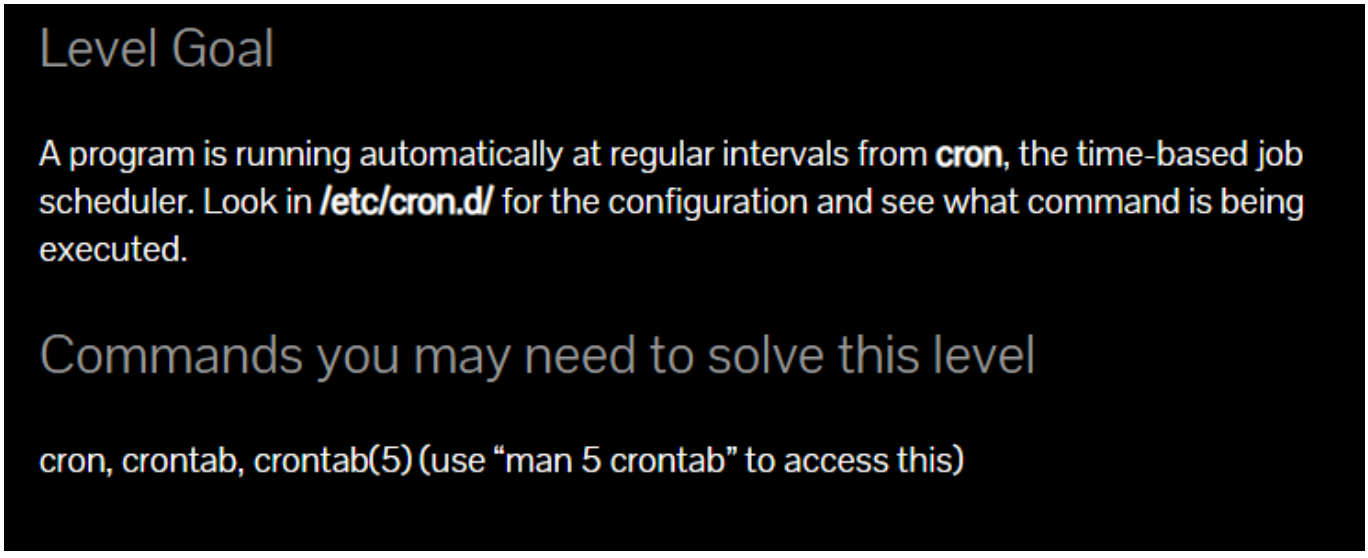
NOTE: Try connecting to your own network daemon to see if it works as you think

For this level we have an executable that makes a connection to localhost on the port we specify, in order to complete this level we must set up.
With `nc -lvnp "port"` we start listening to the port we have sent.
So on the other hand we can execute the setuid file and put the "port" we told nc to listen.



Level 21 → 22

username: bandit21
password: gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr



If we ennumerate we can see that there are some cron commands being executed, we are going to execute one of them and we can see that it is executing an script.

```
bandit21@bandit:~$ ls -al /etc/cron.d
total 36
drwxr-xr-x  2 root root 4096 Jul 11  2020 .
drwxr-xr-x 87 root root 4096 May 14  2020 ..
-rw-r--r--  1 root root   62 May 14  2020 cronjob_bandit15_root
-rw-r--r--  1 root root   62 Jul 11  2020 cronjob_bandit17_root
-rw-r--r--  1 root root  120 May  7  2020 cronjob_bandit22
-rw-r--r--  1 root root  122 May  7  2020 cronjob_bandit23
-rw-r--r--  1 root root  120 May 14  2020 cronjob_bandit24
-rw-r--r--  1 root root   62 May 14  2020 cronjob_bandit25_root
-rw-r--r--  1 root root  102 Oct  7  2017 .placeholder
bandit21@bandit:~$ cat /etc/cron.d/cronjob_bandit22
@reboot bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
* * * * * bandit22 /usr/bin/cronjob_bandit22.sh &> /dev/null
bandit21@bandit:~$
```

```
bandit21@bandit:~$ cat /usr/bin/cronjob_bandit22.sh
#!/bin/bash
chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
bandit21@bandit:~$
```

this script is updating the permissions of the directory /tmp/tmp...
and it is writing the /etc/bandit_pass/bandit22 in /tmp/t....
So finally we open this file.

```
bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI
bandit21@bandit:~$
```

Level 22 → 23

username: bandit22
password:Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

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.

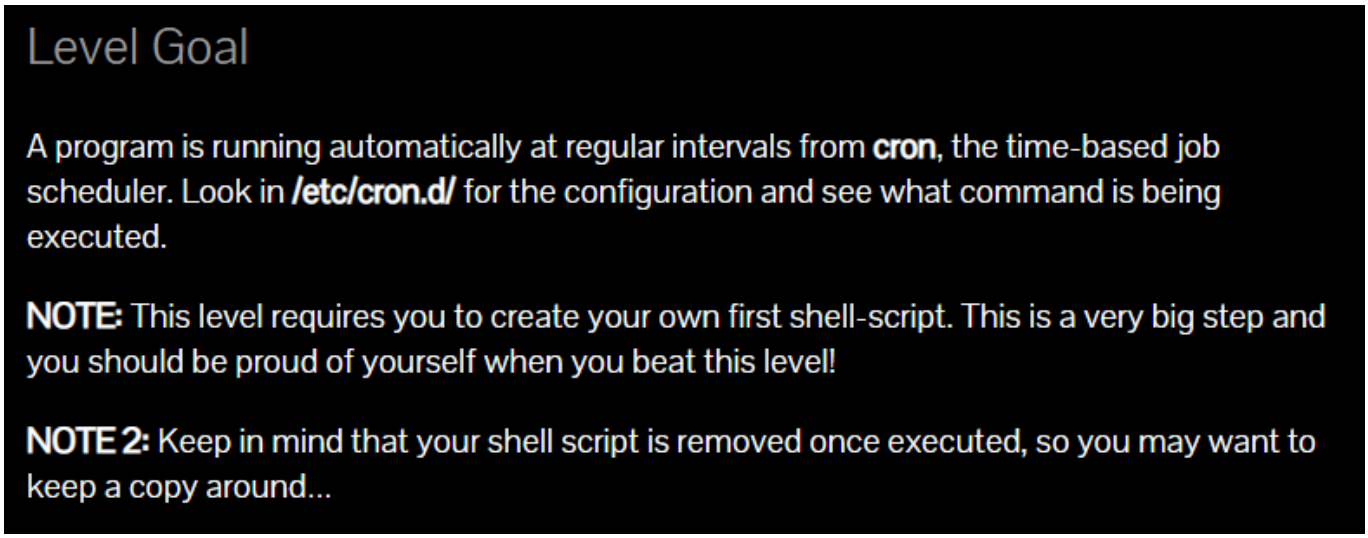
NOTE: Looking at shell scripts written by other people is a very useful skill. The script for this level is intentionally made easy to read. If you are having problems understanding what it does, try executing it to see the debug information it prints.

```
bandit22@bandit:~$ cat /usr/bin/cronjob_bandit23.sh
#!/bin/bash

myname=$(whoami)
mytarget=$(echo I am user $myname | md5sum | cut -d ' ' -f 1)

echo "Copying passwordfile /etc/bandit_pass/$myname to /tmp/$mytarget"

cat /etc/bandit_pass/$myname > /tmp/$mytarget
bandit22@bandit:~$ echo I am user bandit23 | md5sum | cut -d ' ' -f 1
8ca319486bfbbc3663ea0fbe81326349
bandit22@bandit:~$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349
jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n
bandit22@bandit:~$
```

As before we look for the commands that are being executed. We can see the script that is being executed is being executed by **bandit23** so the variable myname is **bandit23**, and then we can execute the next command line in our terminal to guess where is the bandit_pass being copied.

Level 23 → 24

username: bandit23
password: jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n



As before we check what is being executed in /etc/cron.d

```
bandit23@bandit:~$ ls -al /etc/cron.d/
total 36
drwxr-xr-x  2 root root 4096 Jul 11  2020 .
drwxr-xr-x 87 root root 4096 May 14  2020 ..
-rw-r--r--  1 root root   62 May 14  2020 cronjob_bandit15_root
-rw-r--r--  1 root root   62 Jul 11  2020 cronjob_bandit17_root
-rw-r--r--  1 root root  120 May  7  2020 cronjob_bandit22
-rw-r--r--  1 root root  122 May  7  2020 cronjob_bandit23
-rw-r--r--  1 root root  120 May 14  2020 cronjob_bandit24
-rw-r--r--  1 root root   62 May 14  2020 cronjob_bandit25_root
-rw-r--r--  1 root root  102 Oct  7  2017 .placeholder
bandit23@bandit:~$ cat /etc/cron.d/cronjob_bandit24
@reboot bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
* * * * * bandit24 /usr/bin/cronjob_bandit24.sh &> /dev/null
bandit23@bandit:~$ cat /usr/bin/cronjob_bandit24.sh
#!/bin/bash

myname=$(whoami)

cd /var/spool/$myname
echo "Executing and deleting all scripts in /var/spool/$myname:"
for i in * .*;
do
    if [ "$i" != "." -a "$i" != ".." ];
    then
        echo "Handling $i"
        owner="$(stat --format "%U" ./$i)"
        if [ "${owner}" = "bandit23" ]; then
            timeout -s 9 60 ./$i
        fi
        rm -f ./$i
    fi
done
```

We are going to analyze the script being executed.
The variable myname is bandit24, and the scripts executes and deletes al scripts in the directory /var/spool/bandit24.
Then a loop is started for every element in the directory, and meanwhile i != . or .. the variable owner checks who created the script and then execute it, when this is done it removes the script.

After understanding the code we are going to create an script.

```
  GNU nano 2.7.4

#/bin/bash

cat "/etc/bandit_pass/bandit24" > /tmp/tmp.jsYzQuaUDL/pass.txt
```

This simple script created in the directory /tmp/tmp.jsYzQuaUDL is going to be executed by bandit24, so we are copying the password and write it in a file pass.txt in this directory. For allowing bandit24 to execute the script and write the pass.txt in this directory we must change the permissions of both the script and the directory.

```
bandit23@bandit:/tmp/tmp.jsYzQuaUDL$ chmod o+wrx script.sh | chmod o+wr /tmp/tmp.jsYzQuaUDL
```

Finally we copy our script to the directory /var/spool/bandit24 and wait for the flag.

```
bandit23@bandit:/tmp/tmp.jsYzQuaUDL$ cp script.sh /var/spool/bandit24
bandit23@bandit:/tmp/tmp.jsYzQuaUDL$ ls -al
total 1996
drwx---rwx 2 bandit23 root    4096 Jun 24 13:26 .
drwxrws-wt 1 root     root 2031616 Jun 24 13:35 ..
-rw-r--rwx 1 bandit23 root      76 Jun 24 13:21 script.sh
bandit23@bandit:/tmp/tmp.jsYzQuaUDL$ ls
pass.txt  script.sh
bandit23@bandit:/tmp/tmp.jsYzQuaUDL$ cat pass.txt
UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ
```

Level 24 → 25

username: bandit24
password: UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ

Level Goal

A daemon is listening on port 30002 and will give you the password for bandit25 if given the password for bandit24 and a secret numeric 4-digit pincode. There is no way to retrieve the pincode except by going through all of the 10000 combinations, called brute-forcing.

For this level we have to retrieve the password of bandit25 but with a pincode next to it, for this level we are going to create a dictionary that collects all numbers from 0000-9999.

```
#!/bin/bash

for i in {0000..9999}; do
        echo "UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ $i"
done
```

We create an script that creates the dictionary.
And finally submit it to the port specified.

```
bandit24@bandit:/tmp/tmp.UcSpzyOSnP$ cat dictionary.txt | nc localhost 30002 | grep -v "Wrong"
I am the pincode checker for user bandit25. Please enter the password for user bandit24 and the secret pincode on a single line, separated by a space.
Correct!
The password of user bandit25 is uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG

Exiting.
```

Level 25 → 26

username: bandit25
password: uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG

Level Goal

Logging in to bandit26 from bandit25 should be fairly easy... The shell for user bandit26 is not **/bin/bash**, but something else. Find out what it is, how it works and how to break out of it.

Commands you may need to solve this level

ssh, cat, more, vi, ls, id, pwd

```
bandit25@bandit:~$ ls
bandit26.sshkey
bandit25@bandit:~$ file bandit26.sshkey
bandit26.sshkey: PEM RSA private key
bandit25@bandit:~$
```

we have a private RSA key, we are going to try to connect to bandit 26 with it.
We can't connect so we are going to see what type of shell does bandit26 have.

```
bandit25@bandit:~$ cat /etc/passwd | grep bandit26
bandit26:x:11026:11026:bandit level 26:/home/bandit26:/usr/bin/showtext
bandit25@bandit:~$ cat /usr/bin/showtext
#!/bin/sh

export TERM=linux

more ~/text.txt
exit 0
```

Here we see that the terminal we are using is just a more command.
A way to exploit the more command is making our terminal very small, so it doesn't kick us out.

```
--More--(16%)
```

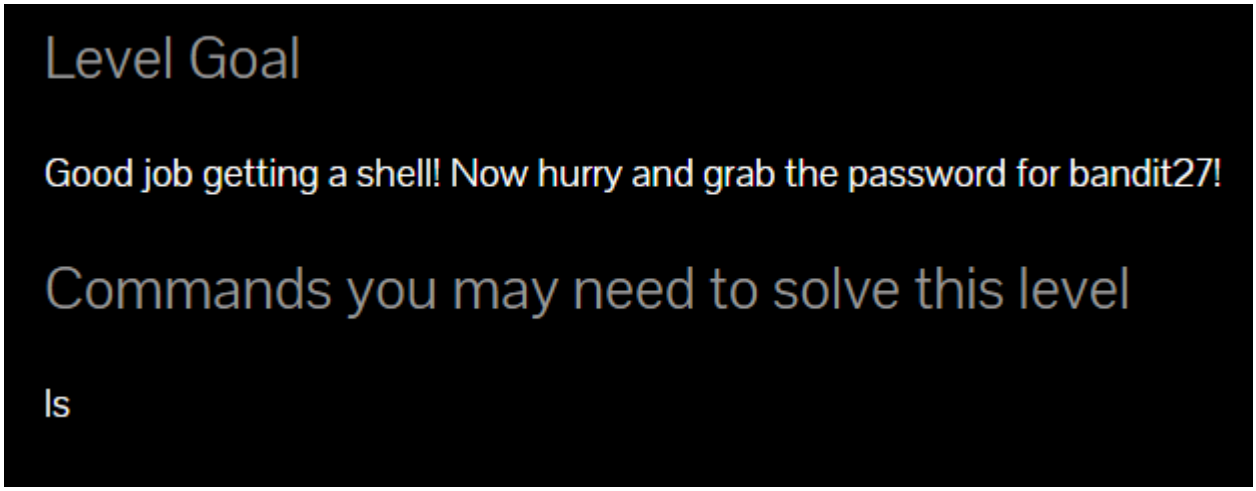
Once we are in we can execute commands clicking `v`

```
~/text.txt[RO] [dec= 95] [hex=5F] [pos=0001:0003][16% of 6]
:set shell=/bin/bash
```

We set up a shell for us and now we use `: shell`

Level 26 → 27

username: bandit 26
password: 5czgV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z



```
bandit26@bandit:~$ ls
bandit27-do  text.txt
bandit26@bandit:~$ ./bandit27-do
Run a command as another user.
  Example: ./bandit27-do id
bandit26@bandit:~$ ./bandit27-do cat "/etc/bandit_pass/bandit27"
3ba3118a22e93127a4ed485be72ef5ea
bandit26@bandit:~$
```

We can see that there is a SUID file and we can execute it and get the next flag

Level 27 → 28

username: bandit27
password: 3ba3118a22e93127a4ed485be72ef5ea

Level Goal

There is a git repository at **ssh://bandit27-git@localhost/home/bandit27-git/repo**. The password for the user **bandit27-git** is the same as for the user **bandit27**.

Clone the repository and find the password for the next level.

Commands you may need to solve this level

git

```
bandit27@bandit:~$ cd /tmp/tmp.A0wbo84GHL
bandit27@bandit:/tmp/tmp.A0wbo84GHL$ git clone ssh://bandit27-git@localhost/home/bandit27-git/repo
Cloning into 'repo'...
Could not create directory '/home/bandit27/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit27/.ssh/known_hosts).
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit27-git@localhost's password:
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (3/3), done.
bandit27@bandit:/tmp/tmp.A0wbo84GHL$ ls
repo
bandit27@bandit:/tmp/tmp.A0wbo84GHL$ cd repo/
bandit27@bandit:/tmp/tmp.A0wbo84GHL/repo$ ls
README
bandit27@bandit:/tmp/tmp.A0wbo84GHL/repo$ cat README
The password to the next level is: 0ef186ac70e04ea33b4c1853d2526fa2
bandit27@bandit:/tmp/tmp.A0wbo84GHL/repo$
```

We clone the repository and read what is inside.

Level 28 → 29

username: bandit28
password: 0ef186ac70e04ea33b4c1853d2526fa2

Level Goal

There is a git repository at **ssh://bandit28-git@localhost/home/bandit28-git/repo**. The password for the user **bandit28-git** is the same as for the user **bandit28**.

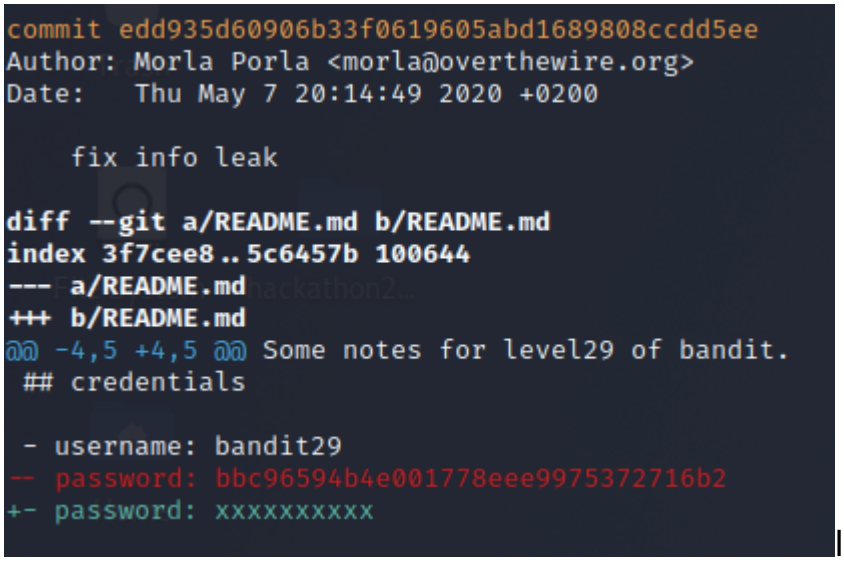
Clone the repository and find the password for the next level.

Commands you may need to solve this level

git

```
bandit28@bandit:~$ mktemp -d
/tmp/tmp.lY1WVKDmpy
bandit28@bandit:~$ cd /tmp/tmp.lY1WVKDmpy
bandit28@bandit:/tmp/tmp.lY1WVKDmpy$ git clone ssh://bandit28-git@localhost/home/bandit28-git/repo
Cloning into 'repo'...
Could not create directory '/home/bandit28/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit28/.ssh/known_hosts).
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit28-git@localhost's password:
Permission denied, please try again.
bandit28-git@localhost's password:
Permission denied, please try again.
bandit28-git@localhost's password:
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 2), reused 0 (delta 0)
Receiving objects: 100% (9/9), 796 bytes | 0 bytes/s, done.
Resolving deltas: 100% (2/2), done.
bandit28@bandit:/tmp/tmp.lY1WVKDmpy$
```

As before we clone the repo and see what is inside README, we see that the password is not there, but we can see the changes with the command `git log -p`

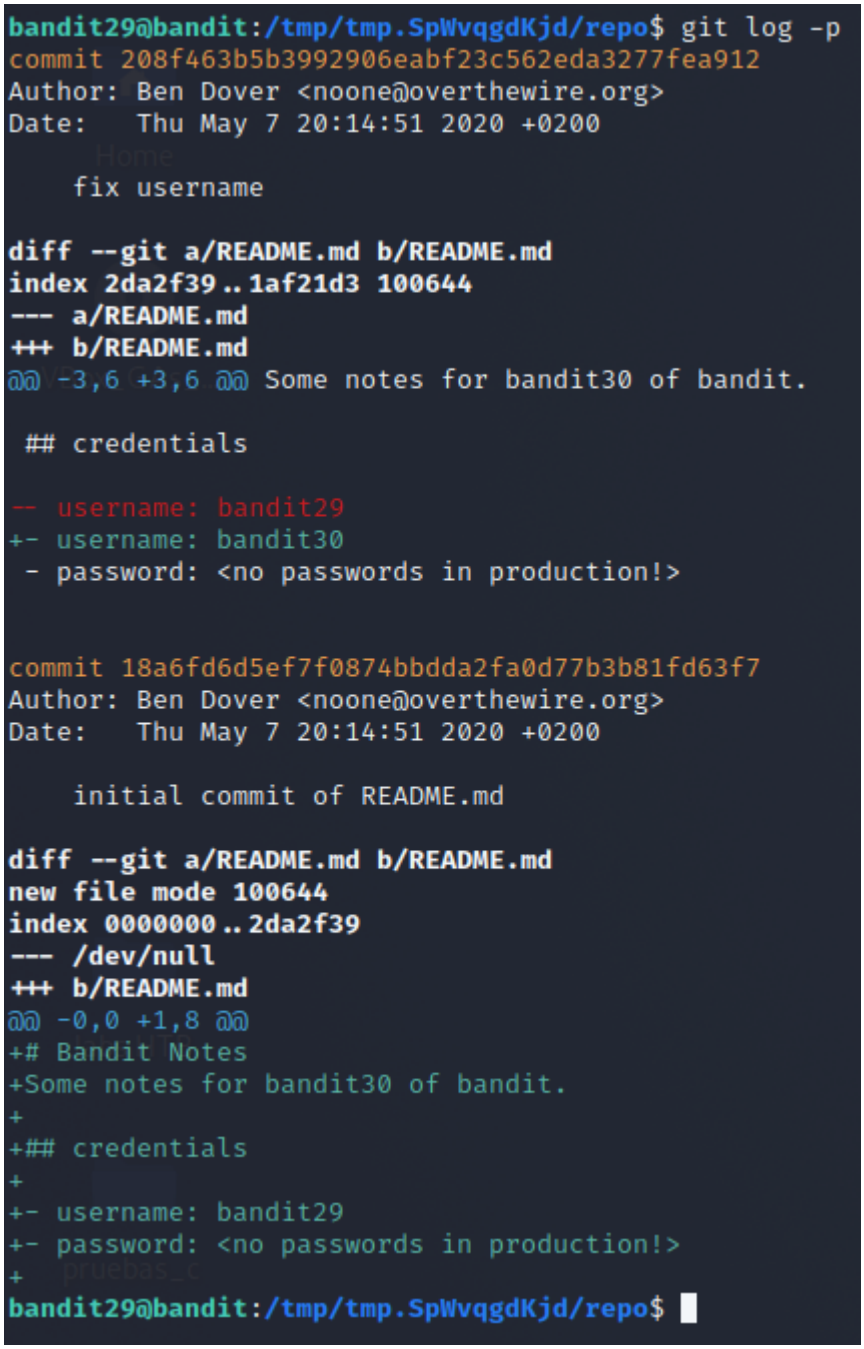


In the changes we can see that they have taken out the password.

Level 29 → 30

username: bandit29
password: bbc96594b4e001778eee9975372716b2

As before we clone the repository



With `git log -p` we can't see any changes, but this is because there are different branches in this repository. We are going to check it with `git branch -r` and then change with `git checkout def`

```
bandit29@bandit:/tmp/tmp.SpWvqgdKjd/repo$ git branch -r
  origin/HEAD → origin/master
  origin/dev
  origin/master
  origin/sploits-dev
bandit29@bandit:/tmp/tmp.SpWvqgdKjd/repo$ git checkout origin/dev
Note: checking out 'origin/dev'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b <new-branch-name>

HEAD is now at bc83328... add data needed for development
bandit29@bandit:/tmp/tmp.SpWvqgdKjd/repo$
```

We see there is a branch with the name "dev" we are going to see his changes.

```
commit bc833286fca18a3948aec989f7025e23ffc16c07
Author: Morla Porla <morla@overthewire.org>
Date:   Thu May 7 20:14:52 2020 +0200

    add data needed for development

diff --git a/README.md b/README.md
index 1af21d3..39b87a8 100644
--- a/README.md
+++ b/README.md
@@ -4,5 +4,5 @@ Some notes for bandit30 of bandit.
 ## credentials

 - username: bandit30
-- password: <no passwords in production!>
+- password: 5b90576bedb2cc04c86a9e924ce42faf
```

Level 30 → 31

username: bandit30
password: 5b90576bedb2cc04c86a9e924ce42faf

Level Goal

There is a git repository at **ssh://bandit30-git@localhost/home/bandit30-git/repo**. The password for the user **bandit30-git** is the same as for the user **bandit30**.

Clone the repository and find the password for the next level.

```
bandit30@bandit:/tmp/tmp.LGd9TUkXcy/repo$ ls
README.md
bandit30@bandit:/tmp/tmp.LGd9TUkXcy/repo$ cat README.md
just an epmty file ... muahaha
bandit30@bandit:/tmp/tmp.LGd9TUkXcy/repo$ git log -p
commit 3aefa229469b7ba1cc08203e5d8fa299354c496b
Author: Ben Dover <noone@overthewire.org>
Date:   Thu May 7 20:14:54 2020 +0200

    initial commit of README.md

diff --git a/README.md b/README.md
new file mode 100644
index 0000000..029ba42
--- /dev/null
+++ b/README.md
@@ -0,0 +1 @@
+just an epmty file ... muahaha
bandit30@bandit:/tmp/tmp.LGd9TUkXcy/repo$ git branch -r
  origin/HEAD -> origin/master
  origin/master
```

As this is the only branch and we can't see any commit that cointains the password we are going to check out for any tags with the command `git tag` and finally we are going to open it with `git show "nametag"`

```
bandit30@bandit:/tmp/tmp.LGd9TUkXcy/repo$ git tag
secret
bandit30@bandit:/tmp/tmp.LGd9TUkXcy/repo$ git show secret
47e603bb428404d265f59c42920d81e5
```

Level 31 → 32

username: bandit31
password: 47e603bb428404d265f59c42920d81e5

Level Goal

There is a git repository at **ssh://bandit31-git@localhost/home/bandit31-git/repo**. The password for the user **bandit31-git** is the same as for the user **bandit31**.

Clone the repository and find the password for the next level.

For this level , after we open the file **README**, we can see that this time we are told to push a file key.txt.

```
bandit31@bandit:/tmp/tmp.gpTD2DnKy1$ cd repo/
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ ls
README.md
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ cat README.md
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

bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ touch key.txt
```

We are going to create it and put the content 'May I come in?'
We add it and then push it.

```
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ git add key.txt
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ git push key.txt
fatal: Invalid gitfile format: key.txt
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ cat .
./  ../  .git/  .gitignore
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ cat .gitignore
*.txt
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ nano .gitignore
```

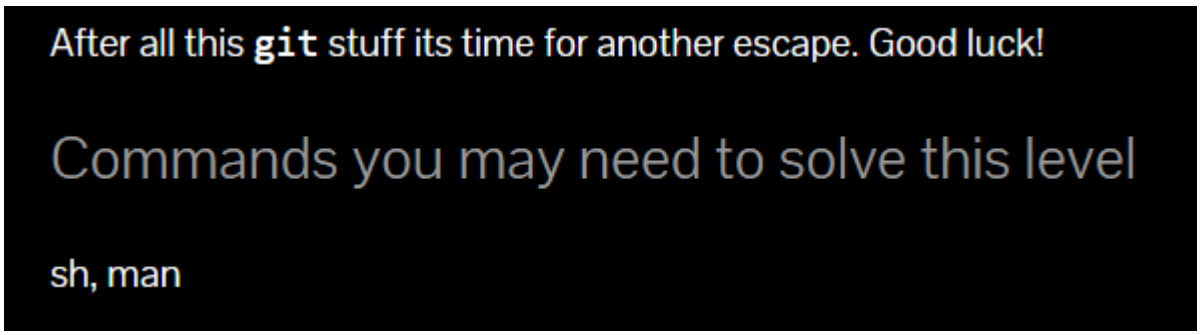
We can see we are not allowed to push it because of the file .gitignore, we are going to remove it. or erase what is in it.

We commit with a message and finally push it

```
bandit31@bandit:/tmp/tmp.gpTD2DnKy1/repo$ git push
Could not create directory '/home/bandit31/.ssh'.
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/bandit31/.ssh/known_hosts).
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit31-git@localhost's password:
Counting objects: 3, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 331 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote: ### Attempting to validate files... ####
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
remote: Well done! Here is the password for the next level:
remote: 56a9bf19c63d650ce78e6ec0354ee45e
remote:
remote: .oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.oOo.
remote:
To ssh://localhost/home/bandit31-git/repo
 ! [remote rejected] master -> master (pre-receive hook declined)
error: failed to push some refs to 'ssh://bandit31-git@localhost/home/bandit31-git/repo'
```

Level 32 → 33

username: bandit32
password: 56a9bf19c63d650ce78e6ec0354ee45e




```
WELCOME TO THE UPPERCASE SHELL
>> whoami
sh: 1: WHOAMI: not found
>> whoami
sh: 1: WHOAMI: not found
>> $0
$
```

In this level everything we put is received in UPPERCASE, so if we execute the command `$0` we are going to spawn a bash. This works because $0 prints what is executing this command, as we are in a terminal bash, it is being executed by a bash, so in conclusion we are doing the same as writting **bash**. Now we can get the password.

```
$ cat /etc/bandit_pass/bandit33
c9c3199ddf4121b10cf581a98d51caee
$
```

Level 33

username: bandit 33
password: c9c3199ddf4121b10cf581a98d51caee

```
bandit33@bandit:~$ ls
README.txt
bandit33@bandit:~$ cat README.txt
Congratulations on solving the last level of this game!

At this moment, there are no more levels to play in this game. However, we are constantly working
on new levels and will most likely expand this game with more levels soon.
Keep an eye out for an announcement on our usual communication channels!
In the meantime, you could play some of our other wargames.

If you have an idea for an awesome new level, please let us know!
```

Congratulations! You are done.