CFSS PROJECT Write-up for the Bandit Wargame

SUBMITTED BY AHATESHAM MOPAGAR

The Bandit wargame is an online game offered by the OverTheWire community. It helps you to learn various Linux commands and understand some basic features of this system.

This is a quick write-up of my solutions for this challenge. I advise you do it yourself before looking at the solutions as you won't learn anything without trying. My goal here is simply to show you how I did it and compare your solutions with mine.

Note: You should follow this write-up with the official website open as it gives details on the goal of each challenges and some helpful material to read.

Bandit 00 Solution

The host to which you need to connect is **bandit.labs.overthewire.org**, on port **2220**. The username is **bandit0** and the password is **bandit0**. The password for the next level is stored in a file called **readme** located in the home directory.

```
$ ssh bandit0@bandit.labs.overthewire.org -p 2220
```

```
$ ls -la
total 24
drwxr-xr-x  2 root    root    4096 Oct 16 14:00 .
drwxr-xr-x 41 root    root    4096 Oct 16 14:00 ..
-rw-r--r--  1 root    root     220 May 15  2017 .bash_logout
-rw-r--r--  1 root    root    3526 May 15  2017 .bashrc
-rw-r--r--  1 root    root     675 May 15  2017 .profile
-rw-r-----  1 bandit1 bandit0   33 Oct 16 14:00 readme
bandit0@bandit:~$ cat readme
boJ9jbbUNNfktd78OOpsqOltutMc3MY1
```

Explanation: Here, you just need to read the content of the **readme** file with the command `cat`.

Bandit 01 Solution

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

```
$ ssh bandit1@bandit.labs.overthewire.org -p 2220
```

```
bandit1@bandit: $ cat ./-
CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9
bandit1@bandit:~$
```

Explanation: As **'-'** means reading from/to stdin in a shell, you need to specify a path to read the file. If you don't specify the path, `cat` will read from *stdin* and print back your input.

Bandit 02 Solution

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

```
$ ssh bandit2@bandit.labs.overthewire.org -p 2220

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

Explanation: You can also read the file by escaping the **spaces** using backslash (**'\'**) like the following command: `cat spaces\ in\ this\ filename`.

Bandit 03 Solution

The password for the next level is stored in a hidden file in the **inhere** directory.

```
$ ssh bandit3@bandit.labs.overthewire.org -p 2220

bandit3@bandit:~$ ls
inhere
bandit3@bandit:~$ cd inhere/
bandit3@bandit:~/inhere$ ls
bandit3@bandit:~/inhere$ ls -la
total 12
drwxr-xr-x 2 root    root    4096 Dec 28 14:34 .
drwxr-xr-x 3 root    root    4096 Dec 28 14:34 ..
-rw-r----- 1 bandit4 bandit3   33 Dec 28 14:34 .hidden
bandit3@bandit:~/inhere$ cat .hidden
pIwrPrtPN36QITSp3EQaw936yaFoFgAB
```

Explanation: In the Linux operating system, a **hidden** file is any file that begins with a **"."**. When a file is hidden it can not been seen with the bare `ls` command. If you need to see hidden files using the `ls` command you need to add the **-a** switch.

Bandit 04 Solution

The password for the next level is stored in the only human-readable file in the **inhere** directory.

```
$ ssh bandit4@bandit.labs.overthewire.org -p 2220

bandit4@bandit:~$ ls
inhere
bandit4@bandit:~$ cd inhere/
bandit4@bandit:~/inhere$ file ./-file0*
./-file00: data
./-file01: data
./-file02: data
./-file03: data
./-file04: data
./-file05: data
./-file06: data
./-file07: ASCII text
./-file08: data
./-file09: data
bandit4@bandit:~/inhere$ cat ./-file07
```

```
koReBOKuIDDepwhWk7jZC0RTdopnAYKh
```

Explanation: Here, we use the `file` command with a *wildcard* on the filename to find the file containing only ASCII text.

Bandit 05 Solution

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

```
$ ssh bandit5@bandit.labs.overthewire.org -p 2220
```

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

Explanation: The `find` command is really useful when you look for a specific file. Here, we use the `-readable`, `! -executable` and `-size 1033c` parameters to find a file with the specified properties.

Bandit 06 Solution

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

```
$ ssh bandit6@bandit.labs.overthewire.org -p 2220
```

```
$ find / -type f -size 33c -group bandit6 -user bandit7 2>&1 | grep -v
"Permission denied"
/var/lib/dpkg/info/bandit7.password
find: ‘/proc/11148/task/11148/fdinfo/6’: No such file or directory
find: ‘/proc/11148/fdinfo/5’: No such file or directory
bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password
HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs
```

Explanation: Same as the previous level except that we redirect the files we cannot read to **stderr**. Also we tell `find` to look into the **root** of the file system as we don’t know where the file is located.

Bandit 07 Solution

The password for the next level is stored in the file **data.txt** next to the word **millionth**.

```
$ ssh bandit7@bandit.labs.overthewire.org -p 2220
```

```
bandit7@bandit:~$ find / -name "data.txt" -exec grep -H 'millionth' {} \; 2>&1 |
grep -v "Permission denied"
/home/bandit7/data.txt:millionth        cvX2JJa4CFALtqS87jk27qwqGhBM9plV
```

Explanation: Here we use the `-exec` argument of `find` with the `grep` command to find the file containing the word **millionth**.

Bandit 08 Solution

The password for the next level is stored in the file **data.txt** and is the only line of text that occurs only once.

```
$ ssh bandit8@bandit.labs.overthewire.org -p 2220

bandit8@bandit:~$ sort data.txt | uniq -c | grep "1 "
      1 UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR
```

Explanation: First we use `sort` to sort alphabetically the data in the **data.txt** file then, we use `uniq` to count the number or occurances and find the line of text that occurs only once.

Bandit 09 Solution

The password for the next level is stored in the file **data.txt** in one of the few human-readable strings, beginning with several ‘=’ characters.

```
$ ssh bandit9@bandit.labs.overthewire.org -p 2220

bandit9@bandit:~$ strings data.txt | grep "^=="
========== password
========== isa
========== truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk
```

Explanation: The `strings` command helps us to find the human-readable strings and then `grep` the strings beginning with several **‘=’** characters.

Bandit 10 Solution

The password for the next level is stored in the file **data.txt**, which contains *base64* encoded data.

```
$ ssh bandit10@bandit.labs.overthewire.org -p 2220

bandit10@bandit:~$ ls
data.txt
bandit10@bandit:~$ cat data.txt
VGhlIHBhc3N3b3JkIGlzIElGdWt3S0dzRlc4TU9xM0lSRnFyeEUxaHhUTkViVVBSCg==
bandit10@bandit:~$ cat data.txt | base64 -d
The password is IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR
```

Explanation: Read the **data.txt** and redirect the output to the `base64` command. The **-d** argument is used to decode the string.

Bandit 11 Solution

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.

```
$ ssh bandit11@bandit.labs.overthewire.org -p 2220

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

Explanation: The `tr` command is used to translate the first set of characters **'A-Za-z'** to **'N-ZA-Mn-za-m'** which is a rotation of 13 positions of the first set.

Bandit 12 Solution

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.

```
$ ssh bandit12@bandit.labs.overthewire.org -p 2220

# Create a working folder
bandit12@bandit:~$ mkdir /tmp/ax
bandit12@bandit:~$ cp data.txt /tmp/ax
bandit12@bandit:~$ cd /tmp/ax
# Convert hexdump to binary
bandit12@bandit:/tmp/ax$ xxd -r data.txt data.out
bandit12@bandit:/tmp/ax$ file data.out
data.out: gzip compressed data, was "data2.bin", last modified: Tue Oct 16
12:00:23 2018, max compression, from Unix
bandit12@bandit:/tmp/ax$ mv data.out data.gz
bandit12@bandit:/tmp/ax$ gzip -d data.gz
bandit12@bandit:/tmp/ax$ file data
data: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/ax$ bzip2 -d data
bzip2: Can\'t guess original name for data -- using data.out
bandit12@bandit:/tmp/ax$ file data.out
data.out: gzip compressed data, was "data4.bin", last modified: Tue Oct 16
12:00:23 2018, max compression, from Unix
bandit12@bandit:/tmp/ax$ mv data.out data.gz
bandit12@bandit:/tmp/ax$ gzip -d data.gz
bandit12@bandit:/tmp/ax$ file data
data: POSIX tar archive (GNU)
bandit12@bandit:/tmp/ax$ tar -xf data
bandit12@bandit:/tmp/ax$ file data5.bin
data5.bin: POSIX tar archive (GNU)
bandit12@bandit:/tmp/ax$ tar -xf data5.bin
bandit12@bandit:/tmp/ax$ file data6.bin
data6.bin: bzip2 compressed data, block size = 900k
bandit12@bandit:/tmp/ax$ bzip2 -d data6.bin
bzip2: Can\'t guess original name for data6.bin -- using data6.bin.out
bandit12@bandit:/tmp/ax$ file data6.bin.out
data6.bin.out: POSIX tar archive (GNU)
bandit12@bandit:/tmp/ax$ tar -xf data6.bin.out
bandit12@bandit:/tmp/ax$ file data8.bin
data8.bin: gzip compressed data, was "data9.bin", last modified: Tue Oct 16
12:00:23 2018, max compression, from Unix
bandit12@bandit:/tmp/ax$ mv data8.bin data8.gz
bandit12@bandit:/tmp/ax$ gzip -d data8.gz
# Finally
bandit12@bandit:/tmp/ax$ file data8
data8: ASCII text
```

```
bandit12@bandit:/tmp/ax$ cat data8
The password is 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL
```

Explanation: The `-r` switch of `xxd` convert an hexdump to binary. Then we use the `file` command to find out which compression tool has been used and recursively decompress the files with the right tool.

Bandit 13 Solution

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.

```
$ ssh bandit13@bandit.labs.overthewire.org -p 2220

bandit13@bandit:~$ ls -la
total 24
drwxr-xr-x  2 root     root     4096 Oct 16 14:00 .
drwxr-xr-x 41 root     root     4096 Oct 16 14:00 ..
-rw-r--r--  1 root     root      220 May 15  2017 .bash_logout
-rw-r--r--  1 root     root     3526 May 15  2017 .bashrc
-rw-r--r--  1 root     root      675 May 15  2017 .profile
-rw-r-----  1 bandit14 bandit13 1679 Oct 16 14:00 sshkey.private
bandit13@bandit:~$ exit
logout
Connection to bandit.labs.overthewire.org closed.

# On your local machine
$ scp -P 2220 bandit13@bandit.labs.overthewire.org:sshkey.private .
$ chmod 400 sshkey.private
$ ssh -i sshkey.private bandit14@bandit.labs.overthewire.org -p 2220

bandit14@bandit:~$
```

Explanation: Here, we download the private key to login to the next level. The `scp` command will do the trick.

Bandit 14 Solution

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

```
$ ssh -i sshkey.private bandit14@bandit.labs.overthewire.org -p 2220

bandit14@bandit:~$ cat /etc/bandit_pass/bandit14 | nc localhost 30000
Correct!
BfMYroe26WYalil77FoDi9qh59eK5xNr
```

Explanation: After login to **bandit14** with the private key, you can redirect the content of **/etc/bandit_pass/bandit14** to netcat using the `nc` command.

Bandit 15 Solution

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.

```
$ ssh bandit15@bandit.labs.overthewire.org -p 2220
```

```
bandit15@bandit:~$ cat /etc/bandit_pass/bandit15 | openssl s_client -connect
localhost:30001 -quiet
depth=0 CN = bandit
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = bandit
verify return:1
Correct!
cluFn7wTiGryunymYOu4RcffSxQluehd
```

Explanation: Here, we send the content of **/etc/bandit_pass/bandit15** to `openssl`. The `s_client` implements a generic SSL/TLS client which can establish a transparent connection to a remote server speaking SSL/TLS.

Bandit 16 Solution

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.

```
$ ssh bandit16@bandit.labs.overthewire.org -p 2220
```

```
bandit16@bandit:~$ for i in {31000..32000} ; do
>   SERVER="localhost"
>   PORT=$i
>   (echo  > /dev/tcp/$SERVER/$PORT) >& /dev/null &&
>    echo "Port $PORT open"
> done
Port 31518 open
Port 31790 open
```

```
bandit16@bandit:~$ cat /etc/bandit_pass/bandit16 | openssl s_client -connect
localhost:31790 -quiet
depth=0 CN = bandit
verify error:num=18:self signed certificate
verify return:1
depth=0 CN = bandit
verify return:1
Correct!
-----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-----
bandit16@bandit:~$ exit
logout
Connection to bandit.labs.overthewire.org closed.
```

Explanation: You can write a simple port scanner in **bash** and try to connect to the open ports with `openssl`.

Bandit 17 Solution

There are 2 files in the homedirectory: **passwords.old** and **passwords.new**. The password for the next level is in **passwords.new** and is the **only** line that has been changed between passwords.old and passwords.new

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

bandit17@bandit:~$ diff passwords.old passwords.new
42c42
< 6vcSC74ROI95NqkKaeEC2ABVMDX9TyUr
---
> kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd
```

Explanation: The `diff` command will compare 2 files line by line and show you the differences.

Bandit 18 Solution

The password for the next level is stored in a file **readme** in the **homedirectory**. Unfortunately, someone has modified .bashrc to log you out when you log in with SSH.

```
$ ssh bandit18@bandit.labs.overthewire.org -p 2220
Byebye !
Connection to bandit.labs.overthewire.org closed.

$ ssh bandit18@bandit.labs.overthewire.org -p 2220 "cat readme"
bandit18@bandit.labs.overthewire.org's password:
IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x
```

Explanation: You can pass the command you want to execute directly to the `ssh` command to bypass the issue.

Bandit 19 Solution

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.

```
$ ssh bandit19@bandit.labs.overthewire.org -p 2220
```

```
bandit19@bandit:~$ ./bandit20-do
Run a command as another user.
  Example: ./bandit20-do id
bandit19@bandit:~$ ./bandit20-do cat /etc/bandit_pass/bandit20
GbKksEFF4yrVs6il55v6gwY5aVje5f0j
```

Explanation: Nothing to explain here, pretty straightforward.

Bandit 20 Solution

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).

```
$ ssh bandit20@bandit.labs.overthewire.org -p 2220
```

```
# Terminal 1
bandit20@bandit:~$ nc -lp 31337 < /etc/bandit_pass/bandit20
gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

# Terminal 2
bandit20@bandit:~$ ./suconnect 31337
Read: GbKksEFF4yrVs6il55v6gwY5aVje5f0j
Password matches, sending next password
```

Explanation: I suggest you open 2 terminals. Set a listener in the first one and try to connect in the second one. The password should appear in your first terninal.

Bandit 21 Solution

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.

```
$ ssh bandit21@bandit.labs.overthewire.org -p 2220
```

```
bandit21@bandit:~$ ls -la /etc/cron.d/
total 24
drwxr-xr-x  2 root root 4096 Oct 16 14:00 .
drwxr-xr-x 88 root root 4096 Oct 16 14:00 ..
-rw-r--r--  1 root root  120 Oct 16 14:00 cronjob_bandit22
-rw-r--r--  1 root root  122 Oct 16 14:00 cronjob_bandit23
-rw-r--r--  1 root root  120 Oct 16 14:00 cronjob_bandit24
-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:~$ cat /usr/bin/cronjob_bandit22.sh
#!/bin/bash
chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv
Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI
```

Explanation: Just read the **cronjob_bandit22.sh** script executed by `cron`. You'll see where the password will be stored.

Bandit 22 Solution

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.

```
$ ssh bandit22@bandit.labs.overthewire.org -p 2220

bandit22@bandit:~$ ls -la /etc/cron.d/
total 24
drwxr-xr-x  2 root root 4096 Oct 16 14:00 .
drwxr-xr-x 88 root root 4096 Oct 16 14:00 ..
-rw-r--r--  1 root root  120 Oct 16 14:00 cronjob_bandit22
-rw-r--r--  1 root root  122 Oct 16 14:00 cronjob_bandit23
-rw-r--r--  1 root root  120 Oct 16 14:00 cronjob_bandit24
-rw-r--r--  1 root root  102 Oct  7  2017 .placeholder
bandit22@bandit:~$ cat /etc/cron.d/cronjob_bandit23
@reboot bandit23 /usr/bin/cronjob_bandit23.sh  &> /dev/null
* * * * * bandit23 /usr/bin/cronjob_bandit23.sh  &> /dev/null
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
8ca319486bfbbc3663ea0fbe81326349  -
bandit22@bandit:~$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349
jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n
```

Explanation: The script tells us that the file where the password will be stored is an md5 hash. You can compute the hash using the `md5sum` command and retrieve the content of the file.

Bandit 23 Solution

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.

```
$ ssh bandit23@bandit.labs.overthewire.org -p 2220

bandit23@bandit:~$ ls -la /etc/cron.d/
total 24
drwxr-xr-x  2 root root 4096 Oct 16 14:00 .
drwxr-xr-x 88 root root 4096 Oct 16 14:00 ..
-rw-r--r--  1 root root  120 Oct 16 14:00 cronjob_bandit22
-rw-r--r--  1 root root  122 Oct 16 14:00 cronjob_bandit23
-rw-r--r--  1 root root  120 Oct 16 14:00 cronjob_bandit24
-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 $i"
        timeout -s 9 60 ./$i
        rm -f ./$i
    fi
done

bandit23@bandit:~$ mkdir /tmp/alex1234
bandit23@bandit:~$ cd /tmp/alex1234
bandit23@bandit:/tmp/alex1234$ vi script.sh

#!/bin/sh
#cat /etc/bandit_pass/bandit24 >> /tmp/alex1234/bandit24pass

bandit23@bandit:/tmp/alex1234$ chmod 777 script.sh
bandit23@bandit:/tmp/alex1234$ cp script.sh /var/spool/bandit24
bandit23@bandit:/tmp/alex1234$ chmod 777 /tmp/alex1234/
# Wait 1 minute
bandit23@bandit:/tmp/alex1234$ ls
bandit24pass  script.sh
bandit23@bandit:/tmp/alex1234$ cat bandit24pass
UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ
```

Explanation: The `cron` script execute and delete all scripts in **/var/spool/bandit24**. We just need to write our own script, copy it in **/var/spool/bandit24** and wait for the result.

Bandit 24 Solution

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.

```
$ ssh bandit24@bandit.labs.overthewire.org -p 2220

# Just so you can keep going...
uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG
```

Note: After multiple attempts, I didn’t found a valid solution yet. Still working on a viable script.

Bandit 25 & 26 Solution

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.

Note: We will solve Bandit 25 & 26 in this section.

```
$ ssh bandit25@bandit.labs.overthewire.org -p 2220

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
```

```
bandit25@bandit:~$ ls
bandit26.sshkey
bandit25@bandit:~$ ssh -i bandit26.sshkey bandit26@localhost
  _                     _ _ _   ___   __
 | |                   | (_) | |__ \ / /
 | |__   __ _ _ __   __| |_| |_   ) / /_
 | '_ \ / _` | '_ \ / _` | | __| / / '_ \
 | |_) | (_| | | | | (_| | | |_ / /| (_) |
 |_.__/ \__,_|_| |_|\__,_|_|\__|____\___/
Connection to localhost closed.
bandit25@bandit:~$

# Reduce the size of the terminal to enable 'more' to paging through text one
screenful at a time.
# Max height = 6


  _                     _ _ _   ___   __
 | |                   | (_) | |__ \ / /
 | |__   __ _ _ __   __| |_| |_   ) / /_
 | '_ \ / _` | '_ \ / _` | | __| / / '_ \
 | |_) | (_| | | | | (_| | | |_ / /| (_) |
--More--(83%)
# Press 'v' to start vi
# Then, in vi type ':e /etc/bandit_pass/bandit26'
5czgV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z
~
~
~
~
"/etc/bandit_pass/bandit26" [readonly] 1L, 33C
```

Now, as we already have a shell using `vi`, we can get the password for level 27.

```
:set shell=/bin/bash
:!ls -la
total 36
drwxr-xr-x  3 root     root     4096 Oct 16 14:00 .
drwxr-xr-x 41 root     root     4096 Oct 16 14:00 ..
-rwsr-x---  1 bandit27 bandit26 7296 Oct 16 14:00 bandit27-do
-rw-r--r--  1 root     root      220 May 15  2017 .bash_logout
-rw-r--r--  1 root     root     3526 May 15  2017 .bashrc
-rw-r--r--  1 root     root      675 May 15  2017 .profile
drwxr-xr-x  2 root     root     4096 Oct 16 14:00 .ssh
-rw-r-----  1 bandit26 bandit26  258 Oct 16 14:00 text.txt
:!./bandit27-do cat /etc/bandit_pass/bandit27
3ba3118a22e93127a4ed485be72ef5ea
```

Explanation: In the first part we figure that the fake shell read a file with `more` and exit. As the content of the file is not long enough, we need to reduce the size of the terminal to enable `more` to paging through text one screenful at a time. Once `more` is running we can type **`v`** to open `vi` and execute command through that tool. Same thing for the second part except the `bandit27-do` command will give us the password.

Bandit 27 Solution

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**.

```
$ ssh bandit27@bandit.labs.overthewire.org -p 2220

bandit27@bandit:~$ mkdir /tmp/repo123
bandit27@bandit:~$ cd /tmp/repo123
bandit27@bandit:/tmp/repo123$ git clone
ssh://bandit27-git@localhost/home/bandit27-git/repo.git/
Cloning into 'repo'...
bandit27-git@localhost 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/repo123$ ls
repo
bandit27@bandit:/tmp/repo123$ cd repo/
bandit27@bandit:/tmp/repo123/repo$ ls
README
bandit27@bandit:/tmp/repo123/repo$ cat README
The password to the next level is: 0ef186ac70e04ea33b4c1853d2526fa2
```

Explanation: You just need to create a temporary folder in **/tmp/** and clone the repo. Inside the repo, you'll find the password.

Bandit 28 Solution

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**.

```
$ ssh bandit28@bandit.labs.overthewire.org -p 2220

bandit28@bandit:~$ mkdir /tmp/repo1337
bandit28@bandit:~$ cd /tmp/repo1337
bandit28@bandit:/tmp/repo1337$ git clone ssh://bandit28-
git@localhost/home/bandit28-git/repo
Cloning into 'repo'...
bandit28-git@localhost 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), done.
Resolving deltas: 100% (2/2), done.
bandit28@bandit:/tmp/repo1337$ ls
repo
bandit28@bandit:/tmp/repo1337$ cd repo/
bandit28@bandit:/tmp/repo1337/repo$ ls
README.md
bandit28@bandit:/tmp/repo1337/repo$ cat README.md
# Bandit Notes
Some notes for level29 of bandit.

## credentials

- username: bandit29
- password: xxxxxxxxxx

bandit28@bandit:/tmp/repo1337/repo$ git log
commit 073c27c130e6ee407e12faad1dd3848a110c4f95
Author: Morla Porla <morla@overthewire.org>
Date:   Tue Oct 16 14:00:39 2018 +0200
   fix info leak

commit 186a1038cc54d1358d42d468cdc8e3cc28a93fcb
Author: Morla Porla <morla@overthewire.org>
Date:   Tue Oct 16 14:00:39 2018 +0200

    add missing data

commit b67405defc6ef44210c53345fc953e6a21338cc7
```

```
Author: Ben Dover <noone@overthewire.org>
Date:   Tue Oct 16 14:00:39 2018 +0200

    initial commit of README.md
bandit28@bandit:/tmp/repo1337/repo$ git checkout
186a1038cc54d1358d42d468cdc8e3cc28a93fcb
Previous HEAD position was 073c27c... fix info leak
HEAD is now at 186a103... add missing data
bandit28@bandit:/tmp/repo1337/repo$ cat README.md
# Bandit Notes
Some notes for level29 of bandit.

## credentials

- username: bandit29
- password: bbc96594b4e001778eee9975372716b2
```

Explanation: You need to create a temporary folder in **/tmp/** and clone the repo. Then, to reveal the password you need to checkout an older commit.

Bandit 29 SolutionPermalink

There is a git repository at **ssh://bandit29-git@localhost/home/bandit29-git/repo**. The password for the user **bandit29-git** is the same as for the user **bandit29**.

```
$ ssh bandit29@bandit.labs.overthewire.org -p 2220

bandit29@bandit:~$ mkdir /tmp/plop123
bandit29@bandit:~$ cd /tmp/plop123
bandit29@bandit:/tmp/plop123$ git clone ssh://bandit29-
git@localhost/home/bandit29-git/repo
Cloning into 'repo'...
bandit29-git@localhost password:

remote: Counting objects: 16, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16 (delta 2), reused 0 (delta 0)
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (2/2), done.
bandit29@bandit:/tmp/plop123$ cd repo/
bandit29@bandit:/tmp/plop123/repo$ cat README.md
# Bandit Notes
Some notes for bandit30 of bandit.

## credentials
```

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

bandit29@bandit:/tmp/plop123/repo$ git branch -r
  origin/HEAD -> origin/master
  origin/dev
  origin/master
  origin/sploits-dev
bandit29@bandit:/tmp/plop123/repo$ git checkout dev
Branch dev set up to track remote branch dev from origin.
Switched to a new branch 'dev'
bandit29@bandit:/tmp/plop123/repo$ cat README.md
# Bandit Notes
Some notes for bandit30 of bandit.

## credentials

- username: bandit30
- password: 5b90576bedb2cc04c86a9e924ce42faf
```

Explanation: You need to create a temporary folder in **/tmp/** and clone the repo. Then, to reveal the password you need to checkout the **dev** branch.

Bandit 30 Solution[Permalink]

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**.

```
$ ssh bandit30@bandit.labs.overthewire.org -p 2220

bandit30@bandit:~$ mkdir /tmp/plop1234
bandit30@bandit:~$ cd  /tmp/plop1234
bandit30@bandit:/tmp/plop1234$ git clone ssh://bandit30-
git@localhost/home/bandit30-git/repo
Cloning into 'repo'...
bandit30-git@localhost password:

remote: Counting objects: 4, done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (4/4), done.
```

```
bandit30@bandit:/tmp/plop1234$ cd repo/
bandit30@bandit:/tmp/plop1234/repo$ ls
README.md
bandit30@bandit:/tmp/plop1234/repo$ cat README.md
just an epmty file... muahaha
bandit30@bandit:/tmp/plop1234/repo$ git tag
secret
bandit30@bandit:/tmp/plop1234/repo$ git show secret
47e603bb428404d265f59c42920d81e5
```

Explanation: You need to create a temporary folder in **/tmp/** and clone the repo. `git show` will display the tag message and the referenced objects to reveal the password.

Bandit 31 SolutionPermalink

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**.

```
$ ssh bandit31@bandit.labs.overthewire.org -p 2220

bandit31@bandit:~$ mkdir /tmp/plop12345
bandit31@bandit:~$ cd /tmp/plop12345
bandit31@bandit:/tmp/plop12345$ git clone ssh://bandit31-
git@localhost/home/bandit31-git/repo
Cloning into 'repo'...
bandit31-git@localhost password:

remote: Counting objects: 4, done.
```

```
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0)
Receiving objects: 100% (4/4), done.
bandit31@bandit:/tmp/plop12345$ cd repo/
bandit31@bandit:/tmp/plop12345/repo$ ls
README.md
bandit31@bandit:/tmp/plop12345/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/plop12345/repo$ echo "May I come
in?">key.txt
bandit31@bandit:/tmp/plop12345/repo$ git add -f key.txt
bandit31@bandit:/tmp/plop12345/repo$ git commit -m
key.txt
[master 1e7c122] key.txt
 1 file changed, 1 insertion(+)
 create mode 100644 key.txt
bandit31@bandit:/tmp/plop12345/repo$ git push origin
master
bandit31-git@localhost password:

Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 320 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'
```

Explanation: You need to create a temporary folder in **/tmp/** and clone the repo. Then, we just follow the instruction in the **README.md**. Push a file called **key.txt**, add the file and push it to the **master** branch.

Bandit 32 SolutionPermalink

After all this git stuff its time for another escape.

```
$ ssh bandit32@bandit.labs.overthewire.org -p 2220

WELCOME TO THE UPPERCASE SHELL
>> ls
sh: 1: LS: not found
>> $0
$ vim

# In vim enter the following command :
# :r /etc/bandit_pass/bandit33

c9c3199ddf4121b10cf581a98d51caee
```

Explanation: Here we get an interactive shell by inserting **$0** in the *fake* shell, then we run `vim` end read the password for the next level.

Bandit 33 Solution (The End)Permalink

This one is not really a challenge as there are no more levels to play in this game. But we can still try to login to check the password we found previously.

```
$ ssh bandit33@bandit.labs.overthewire.org -p 2220

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