Linux Tools Primer

Overthewire

About Overthewire

- Platform that provides an environment to practice hacking
- Bandit is the easiest
- To advance to the next level you need a password, so the goal of each level is to get the password of the next level
- All levels except level 0 are denoted Level i ➜ Level i+1
- We won’t start at level 0.

Goal:

- Try to improve knowledge of linux tools
- Try to complete these challenges without looking things up on the internet, and using only man pages

To connect:

- No account required
- Go to overthewire.org
- Connect via ssh:
- ssh -p 2220 bandit[level no]@bandit.labs.overthewire.org
- ie.
- ssh -p 2220 bandit1@bandit.labs.overthewire.org

Level 0

Connect to ssh:

Host: bandit.labs.overthewire.org

Port: 2220

Username: bandit0

Password: bandit0

root@kali:~# ssh -p 2220 bandit0@bandit.labs.overthewire.org

Level 0 → Level 1

bandit0@bandit:~$ ls

readme

bandit0@bandit:~$ cat readme

boJ9jbbUNNfktd78OOpsqOltutMc3MY1

bandit0@bandit:~$

- You now have the password for the next level so you can login as bandit1 and do the next level

Level 1 ➔ Level 2

root@kali:~# ssh -p 2220 bandit1@bandit.labs.overthewire.org

There is a file called -

bandit1@bandit:~$ cat "-" <- This didn’t work

bandit1@bandit:~$ pwd
/home/bandit1

bandit1@bandit:~$ cd ../
bandit1@bandit:/home$ cat bandit1/-
CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

Level 2 ➜ Level 3

oot@kali:~# ssh -p 2220 bandit2@bandit.labs.overthewire.org

Level 2 ➜ Level 3

bandit2@bandit:~$ ls

spaces in this filename

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

UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK

bandit2@bandit:~$

Level 3 ➜ Level 4

root@kali:~# ssh -p 2220 bandit3@bandit.labs.overthewire.org

Level 3 ➜ Level 4

bandit3@bandit:~$ ls
inhere
bandit3@bandit:~$ cd inhere
bandit3@bandit:~/inhere$ ls

➜ Huh it appears to be empty

Level 3 ➔ Level 4 (cont)

bandit3@bandit:~/inhere$ ls -lai ← View all files
total 12

131296 drwxr-xr-x 2 root root 4096 Oct 16 2018 .

131004 drwxr-xr-x 3 root root 4096 Oct 16 2018 ..

131302 -rw-r----- 1 bandit4 bandit3 33 Oct 16 2018 .hidden

bandit3@bandit:~/inhere$ cat .hidden

pIwrPrtPN36QITSp3EQaw936yaFoFgAB

Level 4 ➔ Level 5

Let’s start here.

Login:

root@kali:~# ssh -p 2220 bandit4@bandit.labs.overthewire.org

Password:
pIwrPrtPN36QITSp3EQaw936yaFoFgAB

Level 4 ➔ Level 5

Challenge:

The password for the next level is stored in the only human-readable file in the **inhere** directory. Tip: if your terminal is messed up, try the “reset” command.

Hint: Look at the Commands you may need to solve this level section, and find out what each of the commands do.

Use man to view the usage of each command.

Ie. root@kali:~# man ls

Level 4 ➔ Level 5 Solution:

Use the file command:

The file command:

file — determine file type

Allows you to specify multiple files in {}

ie.

$ file -s /dev/hda{,1,2,3,4,5,6,7,8,9,10}

Level 4 ➔ Level 5 Solution:

bandit4@bandit:~/inhere$ cd ../
bandit4@bandit:~$ cat inhere/-file00
bandit4@bandit:~$ file inhere/-file00
inhere/-file00: data

Problem; there is 9 files in the folder so to do it this way be impratical

```
bandit4@bandit:~$ file inhere/-file0{0..9}
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
```

Level 5 ➜ Level 6

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit5@bandit.labs.overthewire.org

Level 5 → Level 6

- This challenge is similar to previous one except there is now 19 folders all with many files with different file name patterns.
- How can you search these?

```
bandit5@bandit:~$ ls
inhere
bandit5@bandit:~$ cd inhere
bandit5@bandit:~/inhere$ ls
maybehere00 maybehere05 maybehere10 maybehere15
maybehere01 maybehere06 maybehere11 maybehere16
maybehere02 maybehere07 maybehere12 maybehere17
maybehere03 maybehere08 maybehere13 maybehere18
maybehere04 maybehere09 maybehere14 maybehere19
```

Level 5 → Level 6 Solution

Use find to search for file with properties:
- human-readable
- 1033 bytes in size
- not executable

-type: type of file, f for regular file
-size: size of the file, c for bytes

```
bandit5@bandit:~$ find inhere -type f -size 1033c
inhere/maybehere07/.file2

bandit5@bandit:~$ cat inhere/maybehere07/.file2
DXjZPULLxYr17uwoI01bNLQbtFemEgo7
```

Level 6 ➜ Level 7

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit6@bandit.labs.overthewire.org

Level 6 ➜ Level 7 Solution

Use find:

In man find:
-group gname File belongs to group gname (numeric group ID allowed).
-user uname File is owned by user uname (numeric user ID allowed).
bandit6@bandit:~$ find / -group bandit6 -user bandit7 -type f -size 33c 2>/dev/null

/var/lib/dpkg/info/bandit7.password

bandit6@bandit:~$ cat /var/lib/dpkg/info/bandit7.password

HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs

Level 7 ➔ Level 8

Challenge:

The password for the next level is stored in the file data.txt next to the word millionth (So you must search within the text file)

Login first:

root@kali:~# ssh -p 2220 bandit7@bandit.labs.overthewire.org

Level 7 → Level 8 Solution

bandit7@bandit:~$ ls

data.txt

bandit7@bandit:~$ grep -e "millionth" data.txt

millionth cvX2JJa4CFALtqS87jk27qwqGhBM9plV

Level 8 ➔ Level 9

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit8@bandit.labs.overthewire.org

Level 8 → Level 9 Solution

Use uniq:

UNIQ MAN:

uniq - report or omit repeated lines

USAGE:

uniq [OPTION]... [INPUT [OUTPUT]]

- only takes input from stdin so you must redirect input

-> sort, then count the number of repeated lines, then get the line that begins with 1 (ie. has the count 1)

Level 8 ➜ Level 9 Solution

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

1 UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR

Level 9 ➔ Level 10

Challenge:

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.

(The file data.txt is an executable)

Login first:

root@kali:~# ssh -p 2220 bandit9@bandit.labs.overthewire.org

Level 9 ➜ Level 10 Solution

MAN strings:

strings - print the sequences of printable characters in files

bandit9@bandit:~$ ls

data.txt

bandit9@bandit:~$ file data.txt

data.txt: data

bandit9@bandit:~$ strings data.txt

Level 9 ➜ Level 10 Solution

Then:

```
bandit9@bandit:~$ strings data.txt | grep "="
2========== the
========== password
>t=  yP
rV~dHm=
========== isa
=FQ?P\U
=     F[
pb=x
J;m=
=)$=
========== truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk
iv8!=
```

Level 10 ➜ Level 11

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit10@bandit.labs.overthewire.org

Level 10 ➜ Level 11 Solution

bandit10@bandit:~$ ls
data.txt
bandit10@bandit:~$ cat data.txt

VGhlIHBhc3N3b3JkIGlzIElGdWt3S0dzRlc4TU9xM0lSRnFyeEUxaHhUTkViVVBS
Cg==

bandit10@bandit:~$ base64 -d data.txt

The password is IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR

bandit10@bandit:~$

Level 11 → Level 12

Challenge:

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

-> This is ROT13 (?)

Login first:

root@kali:~# ssh -p 2220 bandit11@bandit.labs.overthewire.org

Level 11 → Level 12

Challenge:

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

-> This is ROT13 (?)

Login first:

root@kali:~# ssh -p 2220 bandit11@bandit.labs.overthewire.org

Level 11 ➔ Level 12 Solution

Use the tr tool:

TR MAN:

tr - translate or delete characters

SYNOPSIS

tr [OPTION]... SET1 [SET2]

ie. translate lower case to uppercase

$cat somefile | tr “[a-z]” “[A-Z]”

Level 11 ➔ Level 12 Solution

bandit11@bandit:~$ cat data.txt | tr "[a-z]" "[n-za-m]" ← Just lowercase

Ghe password is 5Ge8L4drgPEfPx8ugdwuRK8XSP6k2RHu

Both lowercase and uppercase to get the solution:

bandit11@bandit:~$ cat data.txt | tr "[a-z]" "[n-za-m]" | tr "[A-Z]" "[N-ZA-M]"

The password is 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu

bandit11@bandit:~$

Level 12 ➜ Level 13

Challenge:

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 inwhich you can work using mkdir. For example: mkdir /tmp/myname123.Then copy the datafile using cp, and rename it using mv (read themanpages!)

Login first:

root@kali:~# ssh -p 2220 bandit12@bandit.labs.overthewire.org

Level 12 ➔ Level 13 Solution:

This is a long one. It has been compressed about 7 times.

- The hacky way to do this is to find someone else tmp folder and find their solution

The proper way:
bandit12@bandit:~$ mkdir /tmp/th0r

bandit12@bandit:~$ ls

data.txt

bandit12@bandit:~$ cp data.txt /tmp/th0r/ ⇐ First copy the file to the tmp folder you made

Level 12 ➜ Level 13 Solution

bandit12@bandit:~$ cd /tmp/th0r
bandit12@bandit:/tmp/th0r$ ls
data.txt

bandit12@bandit:/tmp/th0r$ file data.txt
data.txt: ASCII text
bandit12@bandit:/tmp/th0r$ cat data.txt

00000000: 1f8b 0808 d7d2 c55b 0203 6461 7461 322e ....[..data2.

00000010: 6269 6e00 013c 02c3 fd42 5a68 3931 4159 bin..<...BZh91AY

Level 12 ➜ Level 13 Solution

File has file signature: 1F 8B

➜ Which is that of gzip
➜ Extensions gz, or tar.gz
➜ So I tried to rename it and extract it, which is wrong. It is a hexdump.

So first step is to reverse hexdump:

bandit12@bandit:/tmp/th0r$ xxd -r data.txt > data1

bandit12@bandit:/tmp/th0r$ file data1

data1: gzip compressed data, was "data2.bin", last modified: Tue Oct 16 12:00:23 2018, max compression, from Unix

Level 12 ➜ Level 13 Solution

bandit12@bandit:/tmp/th0r$ mv data1 data1.tar.gz

bandit12@bandit:/tmp/th0r$ gzip -d data1.tar.gz ⇐ Decompress the gzip archive

bandit12@bandit:/tmp/th0r$ ls

data1.tar data.txt

bandit12@bandit:/tmp/th0r$

Level 12 ➜ Level 13 Solution

bandit12@bandit:/tmp/th0r$ file data1.tar

data1.tar: bzip2 compressed data, block size = 900k

bandit12@bandit:/tmp/th0r$ bzip2 -d data1.tar ⇐ Decompress the bzip2 archive

bzip2: Can't guess original name for data1.tar -- using data1.tar.out

Level 12 ➔ Level 13 Solution

bandit12@bandit:/tmp/th0r$ ls
data1.tar.out data.txt

bandit12@bandit:/tmp/th0r$ file data1.tar.out ⇐ Rename the file before you decompress it

data1.tar.out: gzip compressed data, was "data4.bin", last modified: Tue Oct 16 12:00:23 2018, max compression, from Unix

bandit12@bandit:/tmp/th0r$ mv data1.tar data1.gz

bandit12@bandit:/tmp/th0r$ gzip -d data1.gz ⇐ Decompress the archive

bandit12@bandit:/tmp/th0r$ ls
data1 data.txt

Level 12 ➔ Level 13 Solution

bandit12@bandit:/tmp/th0r$ file data1

data1: POSIX tar archive (GNU)

bandit12@bandit:/tmp/th0r$

bandit12@bandit:/tmp/th0r$ mv data1 data1.tar

bandit12@bandit:/tmp/th0r$ tar -xvf data1.tar ⇐ Extract the file from the archive

data5.bin

bandit12@bandit:/tmp/th0r$ ls

data1.tar data5.bin data.txt

Level 12 ➜ Level 13 Solution

bandit12@bandit:/tmp/th0r$ file data5.bin

data5.bin: POSIX tar archive (GNU)

bandit12@bandit:/tmp/th0r$ mv data5.bin data5.tar

bandit12@bandit:/tmp/th0r$ tar -xvf data5.tar

data6.bin

bandit12@bandit:/tmp/th0r$ ls

data1.tar data5.tar data6.bin data.txt

Level 12 ➜ Level 13 Solution

bandit12@bandit:/tmp/th0r$ file data6.bin

data6.bin: bzip2 compressed data, block size = 900k

bandit12@bandit:/tmp/th0r$ mv data6.bin data6.tar

bandit12@bandit:/tmp/th0r$ bzip2 -d data6.tar

bzip2: Can't guess original name for data6.tar -- using data6.tar.out

Level 12 ➜ Level 13 Solution

bandit12@bandit:/tmp/th0r$ file data6.tar.out

data6.tar.out: POSIX tar archive (GNU)

bandit12@bandit:/tmp/th0r$ mv data6.tar.out data6.tar

bandit12@bandit:/tmp/th0r$ tar -xvf data6.tar

data8.bin

Level 12 ➜ Level 13 Solution

bandit12@bandit:/tmp/th0r$ 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/th0r$ mv data8.bin data8.gz

bandit12@bandit:/tmp/th0r$ gzip -d data8.gz

bandit12@bandit:/tmp/th0r$ ls

data1.tar data5.tar data6.tar data8 data.txt

Level 12 ➜ Level 13 Solution

Finally the end:

bandit12@bandit:/tmp/th0r$ file data8

data8: ASCII text

bandit12@bandit:/tmp/th0r$ cat data8

The password is 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL

bandit12@bandit:/tmp/th0r$

Level 13 ➜ Level 14

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit13@bandit.labs.overthewire.org

Level 13 → Level 14 Solution:

- Use the given private key to login to the next level:

bandit13@bandit:~$ ls -lai

total 24
130932 drwxr-xr-x 2 root root 4096 Oct 16 2018 .
12 drwxr-xr-x 41 root root 4096 Oct 16 2018 ..
130933 -rw-r--r-- 1 root root 220 May 15 2017 .bash_logout
130935 -rw-r--r-- 1 root root 3526 May 15 2017 .bashrc
130934 -rw-r--r-- 1 root root 675 May 15 2017 .profile
131053 -rw-r----- 1 bandit14 bandit13 1679 Oct 16 2018 sshkey.private

Level 13 → Level 14 Solution:

Find the port of the ssh server: (Becasue it isn’t not the same as the one you normally login with)
bandit13@bandit:~$ ssh -i sshkey.private -p 2220 bandit14@localhost

ssh: connect to host localhost port 2220: Connection refused
bandit13@bandit:~$ nmap -PS localhost

PORT STATE SERVICE

22/tcp open ssh

30000/tcp open ndmps

Level 13 ➜ Level 14 Solution:

➜ When you use -i it doesn’t ask for a password

bandit13@bandit:~$ ssh -i sshkey.private bandit14@localhost

➜ Retrieve the password

bandit14@bandit:~$ cat /etc/bandit_pass/bandit14

4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e

bandit14@bandit:~$

Level 14 ➜ Level 15

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit14@bandit.labs.overthewire.org

Level 14 ➜ Level 15 Solution

→ Scan for open ports:

bandit14@bandit:~$ nmap -PS localhost

Not shown: 998 closed ports

PORT STATE SERVICE

22/tcp open ssh

30000/tcp open ndmps

Level 14 ➜ Level 15 Solution

→ Display the password (print to standard out) on the connection at port 30000

bandit14@bandit:~$ echo 4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e | nc localhost 30000

Correct!

BfMYroe26WYalil77FoDi9qh59eK5xNr

Level 15 ➔ Level 16

Challenge:

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

Login first:

root@kali:~# ssh -p 2220 bandit15@bandit.labs.overthewire.org

Level 15 ➔ Level 16 Solution

→ Checking out port 30001

bandit15@bandit:~$ nmap -PS -p30001 localhost

PORT STATE SERVICE

30001/tcp open pago-services1

Nmap done: 1 IP address (1 host up) scanned in 0.04 seconds

bandit15@bandit:~$

Level 15 ➔ Level 16 Solution

Use s_client:

openssl s_client

NAME

openssl-s_client, s_client - SSL/TLS client program

-ign_eof

Inhibit shutting down the connection when end of file is reached in

the input.

Level 15 → Level 16 Solution

bandit15@bandit:~$ echo BfMYroe26WYalil77FoDi9qh59eK5xNr | openssl s_client -connect localhost:30001 -ign_eof

CONNECTED(00000003)
depth=0 CN = localhost
verify error:num=18:self signed certificate
…

Correct!
cluFn7wTiGryunymYOu4RcffSxQluehd
closed
bandit15@bandit:~$

Level 16 ➔ Level 17

Challenge:

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 whichdon’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.

Login first:

root@kali:~# ssh -p 2220 bandit16@bandit.labs.overthewire.org

Level 16 ➔ Level 17 Solution:

bandit16@bandit:~$ nmap -PS -p31000-32000 localhost

PORT STATE SERVICE

31518/tcp filtered unknown

31790/tcp open unknown

Level 16 ➔ Level 17 Solution:

bandit16@bandit:~$ nmap -sV -p31000-32000 localhost

Not shown: 999 closed ports

PORT STATE SERVICE VERSION

31518/tcp filtered unknown

31790/tcp open ssl/unknown ⇐ This is the port because the challenge said the port used ssl

Level 16 ➔ Level 17 Solution:

bandit16@bandit:~$ echo cluFn7wTiGryunymYOu4RcffSxQluehd | openssl s_client -connect localhost:31790 -ign_eof

CONNECTED(00000003)

..

Correct!

-----BEGIN RSA PRIVATE KEY----- ⇐ Gives you a private key which you use to login to the next level

...

Level 16 ➔ Level 17 Solution:

Copied private key and saved to file

Ran chmod 600 [FILE] (private keys must have permissions 500 or 600)

root@kali:~# ssh -i private_key_18 -p 2220 bandit17@bandit.labs.overthewire.org

Level 17 ➜ Level 18

Challenge:

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

NOTE: if you have solved this level and see ‘Byebye!’ when trying to log into bandit18, this is related to the next level, bandit19

Level 17 ➔ Level 18 Solution

Use diff to see the differences between the 2 files:

DIFF MAN:
%< lines from FILE1
%> lines from FILE2
bandit17@bandit:~$ diff passwords.new passwords.old

42c42

< kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd ⇐ This is the password

> hlbSBPAWJmL6WFDb06gpTx1pPButblOA

Level 18 ➜ Level 19

Challenge:

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

Level 18 ➔ Level 19 Solution

→ When I did this challenge I didn’t log out of bandit17, it works either way

→ Use the -t parameter to prevent ssh from spawning /bin/bash, and using bashrc

-t Force pseudo-terminal allocation. This can be used to execute arbitrary screen-based programs on a remote machine, which can be very useful, e.g. when implementing menu services. Multiple -t options force tty allocation, even if ssh has no local tty.

Level 18 ➔ Level 19 Solution

bandit17@bandit:~$ ssh -t bandit18@localhost /bin/sh

$ cat readme

IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x

$

Level 19 ➔ Level 20

Challenge: (read on overthewire, it wouldn’t let me copy it)

Use the setuid binary

-> Run a command as another user.

Log in first:

root@kali:~# ssh -p 2220 bandit19@bandit.labs.overthewire.org

Level 19 → Level 20 Solution

bandit19@bandit:~$ ls

bandit20-do

bandit19@bandit:~$ ./bandit20-do

Run a command as another user.

Example: ./bandit20-do id

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

GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Level 20 ➔ Level 21

Challenge:

There is a setuid binary in the home directory that does the following: it makes a connection to localhost on the port you specify as a command line 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).

Log in first:

root@kali:~# ssh -p 2220 bandit20@bandit.labs.overthewire.org

Level 20 ➔ Level 21 Solution

bandit20@bandit:~$ ls

suconnect

→ Ran strings suconnect to see if the password was in maybe:
→ Found the usage:

bandit20@bandit:~$ strings suconnect

Usage: %s <portnumber>

This program will connect to the given port on localhost using TCP. If it receives the correct password from the other side, the next password is transmitted back.

Level 20 ➜ Level 21 Solution

-> In previous challenges we did echo [PASSWORD] | nc localhost [PORT]

→ This time it needs to run in the background

→ To put a process in the background use &

→ Since the nc command needs to wait for suconnect to be run use -l (It i being connected to is what I really mean)

From the man:

-l listen mode, for inbound connects

Level 20 ➜ Level 21 Solution

bandit20@bandit:~$ echo "GbKksEFF4yrVs6il55v6gwY5aVje5f0j" | nc -l localhost -p 61337 &

[1] 17887

bandit20@bandit:~$ ps aux

...

bandit20 17887 0.0 0.0 6304 1580 pts/31 S 19:16 0:00 nc -l localhost

Level 20 ➜ Level 21 Solution

Run sudoconnect:

bandit20@bandit:~$ ./suconnect 61337

Read: GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Password matches, sending next password

gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

[1]+ Done echo "GbKksEFF4yrVs6il55v6gwY5aVje5f0j" | nc -l localhost -p 61337

Level 21 ➔ Level 22

Challenge:

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.

Log in first:

root@kali:~# ssh -p 2220 bandit21@bandit.labs.overthewire.org

Level 21 ➔ Level 22 Solution

bandit21@bandit:~$ ls /etc/cron.d

atop cronjob_bandit22 cronjob_bandit23 cronjob_bandit24

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

--> The 5 stars means it runs every minute.

Level 21 ➔ Level 22 Solution

bandit21@bandit:~$ cat /usr/bin/cronjob_bandit22.sh

#!/bin/bash

chmod 644 /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

cat /etc/bandit_pass/bandit22 > /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

//You can see that the script is writing the password to the file every minute

bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

Level 22 ➔ Level 23

Challenge:
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.

Log in first:

root@kali:~# ssh -p 2220 bandit22@bandit.labs.overthewire.org

Level 22 ➜ Level 23 Solution

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

---> The script is running every minute again

Level 22 ➜ Level 23 Solution

bandit22@bandit:~$ cat /usr/bin/cronjob_bandit23.sh

#!/bin/bash

myname=$(whoami) <= whoami gets the uname of the acting user

mytarget=$(echo I am user $myname | md5sum | cut -d ' ' -f 1) ⇐ This is getting the md5sum of the string and removing spaces

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

cat /etc/bandit_pass/$myname > /tmp/$mytarget ⇐ The password is written to file

Level 22 ➔ Level 23 Solution

Since I am user bandit22, runnning the script will not work:

bandit22@bandit:~$ whoami

bandit22

bandit22@bandit:/$ sh /usr/bin/cronjob_bandit23.sh

Copying passwordfile /etc/bandit_pass/bandit22 to /tmp/8169b67bd894ddbb4412f91573b38db3 ⇐ It is the wrong password

bandit22@bandit:/$ cd /tmp/8ca319486bfbbc3663ea0fbe81326349

-bash: cd: /tmp/8ca319486bfbbc3663ea0fbe81326349: Not a directory

Level 22 ➔ Level 23 Solution

On my machine:

root@kali:~# echo I am user bandit23 | md5sum | cut -d ' ' -f 1

8ca319486bfbbc3663ea0fbe81326349

Then back:

bandit22@bandit:/$ cat /tmp/8ca319486bfbbc3663ea0fbe81326349

jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n

bandit22@bandit:/$

Level 23 ➔ Level 24

Challenge:
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: This level requires you to create your own first shell-script. This is a very big step and you should be proud of yourself when you beat this level!
NOTE 2: Keep in mind that your shell script is removed once executed, so you may want to keep a copy around...

Log in first:
root@kali:~# ssh -p 2220 bandit23@bandit.labs.overthewire.org

Level 23 ➔ Level 24 Solution:

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

→ The script is running every minute again

Level 23 ➔ Level 24 Solution:

```
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
```

What the script does is it executes all the scripts in var/spool/bandit24 and then deletes them all

Level 23 ➔ Level 24 Solution:

So what we need to do is write a script that will write the password to a file that we can read, BUT we have to do it in less than a minute or it will be deleted.

Level 23 ➔ Level 24 Solution:

bandit23@bandit:~$

⇒ whoami bandit23

bandit23@bandit:~$ ls -lai /usr/bin/cronjob_bandit24.sh

413346 -rwxr-x--- 1 bandit24 bandit23 253 Oct 16 2018 /usr/bin/cronjob_bandit24.sh

bandit23@bandit:~$

--> This script should run bandit23 files as bandit24 before deleting them

Level 23 ➔ Level 24 Solution:

To make sure I knew what I was doing I wrote a test script on my machine first:
Test Script:

```
#!/bin/bash

pass=$(cat /root/got_hash_1)

echo "Password $pass"

echo "$pass" > /tmp/th0r_script

root@kali:~# cat /tmp/th0r_script

6000e084bf18c302eae4559d48cb520c:2hY68a
```

Level 23 ➔ Level 24 Solution:

Now on the machine:

bandit23@bandit:/var/spool/bandit24$ vim th0rtestscript

#!/bin/bash

pass=$(cat /etc/bandit_pass/bandit24)

echo “$pass” > /tmp/th0r_script_pass

Level 23 ➔ Level 24 Solution:

You have to make it executable:

bandit23@bandit:/var/spool/bandit24$ chmod +x th0r_script

bandit23@bandit:/var/spool/bandit24$ ls -la th0r_script

-rwxr-xr-x 1 bandit23 bandit23 93 Dec 27 20:20 th0r_script

- Wait a bit:

bandit23@bandit:/var/spool/bandit24$ cat /tmp/th0r_script_pass

“UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ”

Level 24 ➔ Level 25

Challenge:

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

Log in first:

root@kali:~# ssh -p 2220 bandit21@bandit.labs.overthewire.org

Level 24 ➔ Level 25 Solution

There is no crunch, so you have to use bash

bandit24@bandit:~$ ls

bandit24@bandit:~$ crunch

-bash: crunch: command not found

Dictionary or shell script??

Level 24 ➔ Level 25 Solution

FIguring things out on my machine:

```
root@kali:~# for i in {0000..0010}; do echo $i; done
0000
0001
0002
...
0008
0009
0010
```

Level 24 ➔ Level 25 Solution

To get all 4 digit combinations run:

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

Level 24 ➜ Level 25 Solution

Figuring out the format of the daemon:

bandit24@bandit:~$ echo "UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ 0000" | nc localhost 30002

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.
Wrong! Please enter the correct pincode. Try again.
^C

Level 24 ➔ Level 25 Solution

Sending the daemon all combinations:
bandit24@bandit:~$ for i in {0000..9999}; do echo "UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ $i"; done | nc localhost 30002
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.
Wrong! Please enter the correct pincode. Try again.
Wrong! Please enter the correct pincode. Try again.
...
Correct!
The password of user bandit25 is uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG
Exiting.

Level 25 ➔ Level 26

Challenge:

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.

Log in first:

root@kali:~# ssh -p 2220 bandit25@bandit.labs.overthewire.org

Level 25 ➔ Level 26 Solution

bandit25@bandit:~$ ls

bandit26.sshkey

bandit25@bandit:~$ ssh -i bandit26.sshkey bandit26@localhost

→ The shell logs you in and then immediately logs you back out.

→ You have to figure out a way to bypass it.

Level 25 → Level 26 Solution

Figuring out which shell bandit26 is using:

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

Level 25 ➔ Level 26 Solution

What is more?

MAN more

more - file perusal filter for crt viewing (Scrolling for large amounts of text on small terminal window.

--> You can execute a command:

!command or :!command

Execute command in a subshell.

Level 25 ➔ Level 26 Solution

Shrink the window really small so more is used.

press v to enter vim

enter :e /etc/bandit_pass/bandit26

Password for level 26: 5czgV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z

Press v again, and then run

:set shell=/bin/bash

:shell

Level 25 ➔ Level 26 Solution

Shrink the window really small so more is used.

press v to enter vim

enter :e /etc/bandit_pass/bandit26

Password for level 26: 5czgV9L3Xx8JPOyRbXh6lQbmIOWvPT6Z

Press v again, and then run

:set shell=/bin/bash

:shell ⇐ IN order to get a regular bash shell

Level 26 → Level 27

Don’t close the shell you now have

Challenge:

Good job getting a shell! Now hurry and grab the password for bandit27!

Level 26 → Level 27 Solution

bandit26@bandit:~$ id

uid=11026(bandit26) gid=11026(bandit26) groups=11026(bandit26)

bandit26@bandit:~$ strings bandit27-do

Run a command as another user.

Example: %s id

bandit26@bandit:~$ ./bandit27-do cat /etc/bandit_pass/bandit27

3ba3118a22e93127a4ed485be72ef5ea

Level 27 ➜ Level 28

Challenge:

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.

Login first.

root@kali:~# ssh -p 2220 bandit27@bandit.labs.overthewire.org

Level 27 ➜ Level 28 Solution

bandit27@bandit:~$ mkdir /tmp/th0r-27

bandit27@bandit:~$ cd /tmp/th0r-27

bandit27@bandit:/tmp/th0r-27$ git clone ssh://bandit27-git@localhost/home/bandit27-git/repo

Cloning into 'repo'...

Level 27 → Level 28 Solution

bandit27@bandit:/tmp/th0r-27$ cd repo

bandit27@bandit:/tmp/th0r-27/repo$ ls

README

bandit27@bandit:/tmp/th0r-27/repo$ cat README

The password to the next level is: 0ef186ac70e04ea33b4c1853d2526fa2

bandit27@bandit:/tmp/th0r-27/repo$

Level 28 → Level 29

Challenge:

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.

Login first.

root@kali:~# ssh -p 2220 bandit28@bandit.labs.overthewire.org

Level 28 → Level 29 Solution

bandit28@bandit:~$ mkdir /tmp/th0r-28

bandit28@bandit:~$ cd /tmp/th0r-28

bandit28@bandit:/tmp/th0r-28$ git clone ssh://bandit28-git@localhost/home/bandit28-git/repo

Cloning into 'repo'...

Level 28 → Level 29 Solution

bandit28@bandit:/tmp/th0r-28$ cd repo
bandit28@bandit:/tmp/th0r-28/repo$ ls
README.md
bandit28@bandit:/tmp/th0r-28/repo$ cat README.md

Bandit Notes
Some notes for level29 of bandit.
credentials

- username: bandit29
- password: xxxxxxxxxx --> You have to find the previous version of the file

Level 28 → Level 29 Solution

```
bandit28@bandit:/tmp/th0r-28/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/th0r-28/repo$
```

Level 28 → Level 29 Solution

-p option shows the diff after each commit
-1 to show only the last entry

bandit28@bandit:/tmp/th0r-28/repo$ git log -p -1
commit 073c27c130e6ee407e12faad1dd3848a110c4f95
...
diff --git a/README.md b/README.md
index 3f7cee8..5c6457b 100644
--- a/README.md
+++ b/README.md
@@ -4,5 +4,5 @@ Some notes for level29 of bandit.
credentials
- username: bandit29
-- password: bbc96594b4e001778eee9975372716b2

Level 29 → Level 30

Challenge:

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.

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

Login first.

root@kali:~# ssh -p 2220 bandit29@bandit.labs.overthewire.org

Level 29 → Level 30 Solution

bandit29@bandit:~$ mkdir /tmp/th0r-29
bandit29@bandit:~$ cd /tmp/th0r-29
bandit29@bandit:/tmp/th0r-29$ git clone
ssh://bandit29-git@localhost/home/bandit29-git/repo

bandit29@bandit:/tmp/th0r-29$ ls
repo

bandit29@bandit:/tmp/th0r-29$ cd repo/
bandit29@bandit:/tmp/th0r-29/repo$ ls
README.md

Level 29 → Level 30 Solution

bandit29@bandit:/tmp/th0r-29/repo$ cat README.md

Bandit Notes

Some notes for bandit30 of bandit.

credentials

- username: bandit30

- password: <no passwords in production!> ⇒ The password must be a different branch

bandit29@bandit:/tmp/th0r-29/repo$

Level 29 → Level 30 Solution

See if there are other branches:

bandit29@bandit:/tmp/th0r-29/repo/.git$ git branch

* master

bandit29@bandit:/tmp/th0r-29/repo/.git$ git branch -r

origin/HEAD -> origin/master
origin/dev
origin/master
origin/sploits-dev
bandit29@bandit:/tmp/th0r-29/repo/.git$

Level 29 → Level 30 Solution

bandit29@bandit:/tmp/th0r-29/repo$ git checkout dev
Branch dev set up to track remote branch dev from origin.
Switched to a new branch 'dev'
bandit29@bandit:/tmp/th0r-29/repo$ ls
code README.md
bandit29@bandit:/tmp/th0r-29/repo$ cat README.md
Bandit Notes
Some notes for bandit30 of bandit.
credentials
- username: bandit30
- password: 5b90576bedb2cc04c86a9e924ce42faf

Level 30 → Level 31

Challenge:

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.

Login first.

root@kali:~# ssh -p 2220 bandit30@bandit.labs.overthewire.org

Level 30 → Level 31 Solution

bandit30@bandit:~$ mkdir /tmp/th0r-30
bandit30@bandit:~$ cd /tmp/th0r-30
bandit30@bandit:/tmp/th0r-30$ git clone
ssh://bandit30-git@localhost/home/bandit30-git/repo

bandit30@bandit:/tmp/th0r-30$ cd repo
bandit30@bandit:/tmp/th0r-30/repo$ ls

README.md

bandit30@bandit:/tmp/th0r-30/repo$ cat README.md

just an epmty file... muahaha

Level 30 ➔ Level 31 Solution

bandit30@bandit:/tmp/th0r-30/repo/.git$ cd ../
bandit30@bandit:/tmp/th0r-30/repo$ git tag
secret
bandit30@bandit:/tmp/th0r-30/repo$ git show secret
47e603bb428404d265f59c42920d81e5
bandit30@bandit:/tmp/th0r-30/repo$

Level 31 → Level 32

Challenge:

This time your task is to push a file to the remote repository.

Login first.

root@kali:~# ssh -p 2220 bandit31@bandit.labs.overthewire.org

Level 31 → Level 32 Solution

bandit31@bandit:~$ mkdir /tmp/th0r-31
bandit31@bandit:~$ cd /tmp/th0r-31
bandit31@bandit:/tmp/th0r-31$ git clone
ssh://bandit31-git@localhost/home/bandit31-git/repo

bandit31@bandit:/tmp/th0r-31/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

Level 31 → Level 32 Solution

```
bandit31@bandit:/tmp/th0r-31/repo$ vim key.txt
bandit31@bandit:/tmp/th0r-31/repo$ cat key.txt
May I come in?
bandit31@bandit:/tmp/th0r-31/repo$ rm .gitignore
bandit31@bandit:/tmp/th0r-31/repo$ git add key.txt
bandit31@bandit:/tmp/th0r-31/repo$ git commit -m "may i come in?"
[master cc5969d] may i come in?
…
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.
```

Level 32 → Level 33

Challenge:

WELCOME TO UPPERCASE (S)HELL

Level 32 → Level 33 Solution

Logging in:

WELCOME TO THE UPPERCASE SHELL

>> ls

sh: 1: LS: not found

>> LS

sh: 1: LS: not found

>>

Level 32 ➔ Level 33 Solution

Log back into 31:

bandit31@bandit:~$ cat /etc/passwd | grep bandit32

bandit32:x:11032:11032:bandit level 32:/home/bandit32:/home/bandit32/uppershell

bandit31@bandit:~$

Level 32 ➔ Level 33 Solution

---> Uppershell takes the input converts it to uppercase and then runs it

→ $0 is the first parameter

→ ie. $0 = sh

→ $1 = [cmd]

$0 $1 -> sh [cmd]

If you just do $0 you can run commands normallly

Level 32 ➜ Level 33 Solution

```
>> $0
$ pwd
/home/bandit32
$ ls
uppershell
$ cat uppershell
$ id
uid=11033(bandit33) gid=11032(bandit32) groups=11032(bandit32)
$
$ cat /etc/bandit_pass/bandit33
c9c3199ddf4121b10cf581a98d51caee
```

END.

There is no next level.

This site offers many other challenges though!

This took me 2 days to do. I doubt we will get here in 2 hours. Lol. :]