Bandit

level 0 :

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

password : bandit0

---

level 0-1 : the password is in a readme file in the home directory.

commands used : ls and cat

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

password : boJ9jbbUNNfktd78OOpsqOltutMc3MY1

---

level 1-2 : the password is in the - file in the home directory

commands used : cat < -

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

password : CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

---

level 2-3 : the password is in the file named “spaces in this filename”

commands used : cat “spaces in this filename”

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

password : UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK

---

level 3-4 : the password is stored in a hidden file in the inhere directory

commands used : cd inhere , ls -a

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

password : pIwrPrtPN36QITSp3EQaw936yaFoFgAB

---

level 4-5 : the password is stored in the only human-readable file in the inhere directory

commands used :

```
ls
find . -type -f | xargs file
cat ./-file07
man xargs
```

(“.” means current directory)

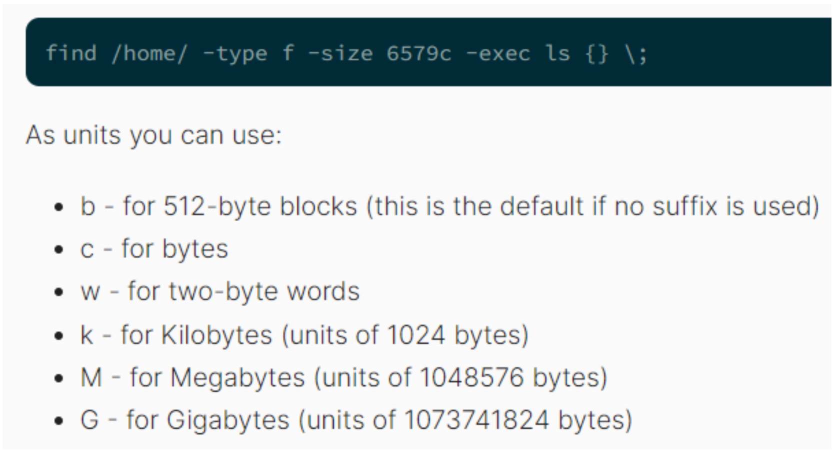
ssh bandit5@bandit.labs.overthewire.org -p 2220

password : koReBOKuIDDepwhWk7jZC0RTdopnAYKh

---

level 5-6 : the password is stored in the inhere directory file has all of the following properties :

- human-readable (didn’t use but -readable)
- 1033 bytes in size (-size 1033c //c for bytes)
- not executable (-type f ! -executable)



```
find /home/ -type f -size 6579c -exec ls {} \;
```

As units you can use:

- b - for 512-byte blocks (this is the default if no suffix is used)
- c - for bytes
- w - for two-byte words
- k - for Kilobytes (units of 1024 bytes)
- M - for Megabytes (units of 1048576 bytes)
- G - for Gigabytes (units of 1073741824 bytes)

command ⇒ find . -type f -size 1033c ! -executable

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

password : DXjZPULLxYr17uwoI01bNLQbtFemEgo7

---

level 6-7 : the password is stored somewhere on the server and has all of the following properties :

- owned by user bandit7 (-user bandit7)
- owned by group bandit6 (-group bandit6)
- 33 bytes in size (-size 33c)

command ⇒ find / -user bandit7 -group bandit6 -size 33c

(/ starts search from the root folder and then all the others (I think))

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

password : HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs

---

level 7-8 : the password is stored in the file **data.txt** next to the word **millionth**

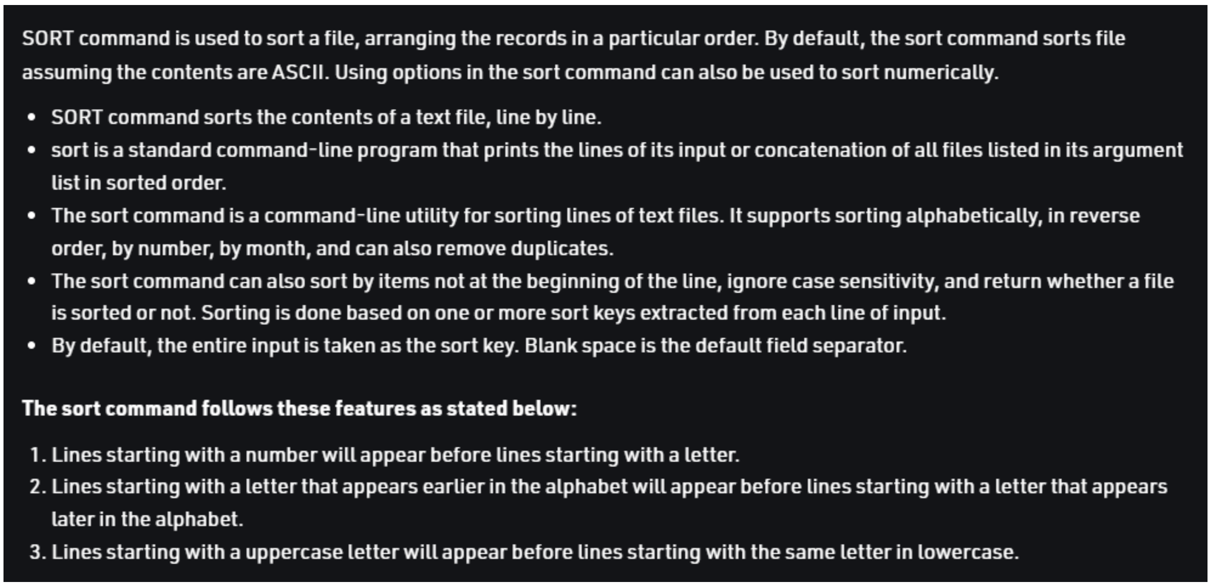
command ⇒ cat data.txt | grep “millionth”

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

password : cvX2JJa4CFALtqS87jk27qwqGhBM9plV

---

level 8-9 : the password is stored in the file **data.txt** and is the only line of text that occurs only once

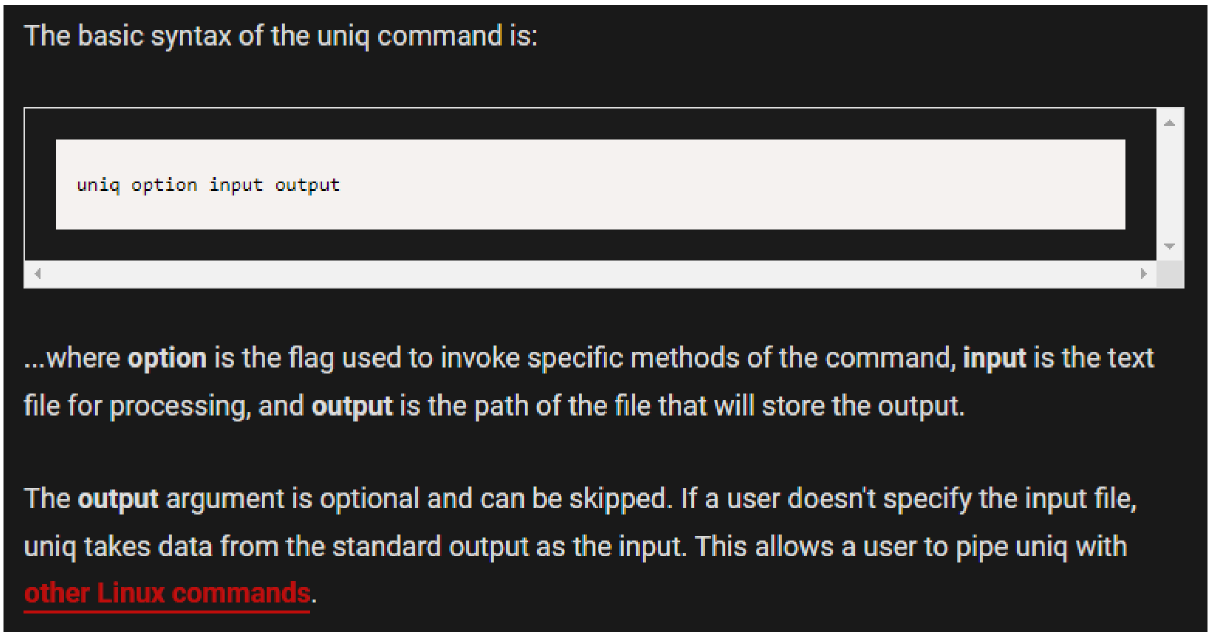

SORT command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ASCII. Using options in the sort command can also be used to sort numerically.

- SORT command sorts the contents of a text file, line by line.
- sort is a standard command-line program that prints the lines of its input or concatenation of all files listed in its argument list in sorted order.
- The sort command is a command-line utility for sorting lines of text files. It supports sorting alphabetically, in reverse order, by number, by month, and can also remove duplicates.
- The sort command can also sort by items not at the beginning of the line, ignore case sensitivity, and return whether a file is sorted or not. Sorting is done based on one or more sort keys extracted from each line of input.
- By default, the entire input is taken as the sort key. Blank space is the default field separator.

The sort command follows these features as stated below:

1. Lines starting with a number will appear before lines starting with a letter.
2. Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet.
3. Lines starting with a uppercase letter will appear before lines starting with the same letter in lowercase.

```
(The uniq command in Linux is used to display identical lines in a text file. This
command can be helpful if you want to remove duplicate words or strings from a text
file. Since the uniq command matches adjacent lines for finding redundant copies, it
only works with sorted text files.)
```


The basic syntax of the uniq command is:

```
uniq option input output
```

...where **option** is the flag used to invoke specific methods of the command, **input** is the text file for processing, and **output** is the path of the file that will store the output.

The **output** argument is optional and can be skipped. If a user doesn't specify the input file, uniq takes data from the standard output as the input. This allows a user to pipe uniq with other Linux commands.

```
command ⇒ sort data.txt | uniq -u

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

password : UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR
```

```
level 9-10 : the password is stored in the file data.txt in one of the few human-
readable strings, preceded by several ‘=’ characters.
```

Linux strings command is used to return the **string characters into files**. It primarily focuses on determining the contents of and extracting text from the binary files (non-text file).

It is a complex task for a human to find out text from an executable file. The binary files, such as program files, contain human-readable text. These files are large-sized if we use a cat or less command; it may cause the terminal to hang up.

There can be two types of characters in a file; printable and non-printable. The alphanumeric characters, punctuation, or whitespaces are known as printable characters; except the printable character, all the characters are known as non-printable characters.

command ⇒ strings data.txt | grep "="

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

password : truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk

---

level 10-11 : the password is in the file **data.txt**, which contains base64 encoded data

command ⇒ cat data.txt | base64 -d

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

password : IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR

---

level 11-12 : the password is stored in the file **data.txt**, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions

used rot13.com (cheated)

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

password : 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu

---

level 12-13 : the password 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 used ⇒

- **xxd -r data.txt >data1** #to revert from hexdump
- made a new dir
- file data1 #to check the filetype (it was gzip compressed)
- mv data1 to data.gz #changed extention to gz
- gzip -d data.gz #-d to decompress
- file data #new file was compressed in bzip2
- mv data data.bz2
- bzip2 -d data.bz2
- file data #new file was compressed in tar
- mv data data.tar
- tar xf data.tar #xf for extract file
- do this multiple times until you get back ASCII text

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

password : 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL

---

level 13-14 : the password 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

command used ⇒ ssh -i sshkeey.private bandit14@localhost

password : no password needed

---

level 14-15 : the password for the next level can be retrieved by submitting the password of the current level to **port 30000 on localhost.**

command used ⇒ telnet localhost 30000

can also use ⇒ nc localhost 30000

current password : 4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e

(stored in /etc/bandit_pass/bandit14)

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

password : BfMYroe26WYalil77FoDi9qh59eK5xNr

---

level 15-16 : 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…

command used ⇒ echo BfMYroe26WYalil77FoDi9qh59eK5xNr | openssl s_client -quiet -connect localhost:30001

(can do it without the echo and quiet too)

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

password : cluFn7wTiGryunymYOu4RcffSxQluehd

---

level 16-17 : the password 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.

to find the correct port, just nmap scan 127.0.0.1 and try

ncat 127.0.0.1 --ssl <port>

p = 31790

once you find the port, save the key in a file in the tmp folder and chmod to 700 (only accessible by bandit16)

ssh bandit17@localhost -i privyet.key

password :

---

level 17-18 : 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

diff passwords.old passwords.new

1 → w0Yfolrc5bwjS4qw5mq1nnQi6mF03bii

2 → kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd