**Level 0:**

**Goal:**

The goal of this level is for you to log into the game using **SSH**. 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**. Once logged in, go to the Level 1 page to find out how to beat Level 1.

**Solution:**

It is not a challenge!

In this level, we are provided the **SSH** connection settings and the initial user to login to the **OvertheWire** server and start [**Bandit wargame**](https://overthewire.org/wargames/bandit/).

**Level 0 - Level 1:**

**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.

**Solution:**

Login to **OvertheWire** server as **bandit0** user by using a **ssh** tool such as [PuTTY](https://www.putty.org/).

Display the contents of the **readme** file.

bandit0@bandit:~$ cat readme

boJ9jbbUNNfktd78OOpsqOltutMc3MY1

The password for the **bandit1** user has been found.

**Level 1 - Level 2:**

**Goal:**

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

**Solution:**

Password is located in **'-'** file. But to display its content is a little tricky task, because regular syntax do not work.

Therefore, we need to provide either the absolute or relative path to open this file.

bandit1@bandit:~$ cat /home/bandit1/-

CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9

We have got the password of **bandit2** user.

**Level 2 - Level 3:**

**Goal:**

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

**Solution:**

We can print the contents of the file as follows.

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

UmHadQclWmgdLOKQ3YNgjWxGoRMb5luK

Password of **bandit3** user has been retrieved.

**Level 3 - Level 4:**

**Goal:**

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

**Solution:**

List the files available in **inhere** directory.

bandit3@bandit:~$ ls -al inhere/

total 12

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

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

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

Print the contents of the one and only file in **inhere** directory.

bandit3@bandit:~$ cat inhere/.hidden

pIwrPrtPN36QITSp3EQaw936yaFoFgAB

**Level 4 - Level 5:**

**Goal:**

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.

**Solution:**

List the contents of **inhere** directory.

bandit4@bandit:~$ ls inhere/

-file00 -file02 -file04 -file06 -file08

-file01 -file03 -file05 -file07 -file09

To find the **human-readable** file, we can use **file** command to check type of each file as follows.

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

**-file07** is our required file. Print its content now to retrieve the password for next level.

bandit4@bandit:~$ cat inhere/-file07

koReBOKuIDDepwhWk7jZC0RTdopnAYKh

Password of **bandit5** user has been located.

**Level 5 - Level 6:**

**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

**Solution:**

Find the files of **1033 bytes** in size within **inhere** directory.

bandit5@bandit:~$ find ./inhere -size 1033c \! -executable

./inhere/maybehere07/.file2

There is only one file of this size. Print its content now.

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

DXjZPULLxYr17uwoI01bNLQbtFemEgo7

We have found the password of **bandit6** user.

**Level 6 - Level 7:**

**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

**Solution:**

Scan the **root (/)** directory to find the files with required conditions.

bandit6@bandit:~$ find / -user bandit7 -group bandit6 -size 33c 2> /dev/null

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

Passwordfile has been found. Print its content now and retrieve the password for next level.

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

HKBPTKQnIay4Fw76bEy8PVxKEDQRKTzs

**Level 7 - Level 8:**

**Goal:**

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

**Solution:**

Use **grep** command to find the row containing word **‘millionth’**.

bandit7@bandit:~$ grep millionth data.txt

millionth cvX2JJa4CFALtqS87jk27qwqGhBM9plV

Password of **bandit8** user is written next to the **‘millionth’** word. Use this password to login as **bandit8** user.

**Level 8 - Level 9:**

**Goal:**

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

**Solution:**

To extract the following line of text, we need to combine multiple commands together as follows.

bandit8@bandit:~$ sort data.txt | uniq -c | sort | tail -1

1 UsvVyFSfZZWbi6wgC7dAFyFuR6jQQUhR

We have located the password for next level.

**Level 9 - Level 10:**

**Goal:**

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.

**Solution:**

Use the **strings** command to list down the human-readable strings in **data.txt** file, then use a grep command to filter strings that begin with several **'='** characters.

bandit9@bandit:~$ strings data.txt | grep '^==='

========== password

========== isa

========== truKLdjsbJ5g7yyJ2X2R0o3a5HQJFuLk

**Level 10 - Level 11:**

**Goal:**

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

**Solution:**

Use **base64** command to decode the text in **data.txt** file.

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

The password is IFukwKGsFW8MOq3IRFqrxE1hxTNEbUPR

Use this password to login to next level.

**Level 11 - Level 12:**

**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

**Solution:**

Text in **data.txt** file has been encrypted using [ROT13](https://en.wikipedia.org/wiki/ROT13) algorithm. We can use the **tr** command to decrypt it.

bandit11@bandit:~$ cat data.txt | tr 'A-Ma-mN-Zn-z' 'N-Zn-zA-Ma-m'

The password is 5Te8Y4drgCRfCx8ugdwuEX8KFC6k2EUu

**Level 12 - Level 13:**

**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!)

**Solution:**

Create a temporary directory and copy**data.txt** file therein.

bandit12@bandit:~$ mkdir /tmp/bandit12\_ahmer

bandit12@bandit:~$ cd /tmp/bandit12\_ahmer

bandit12@bandit:/tmp/bandit12\_ahmer$ cp ~/data.txt .

It is given that the **data.txt** file is the **hexdump** of another file. Therefore, use **xxd** command with reverse flag to obtain the original file.

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

Check the file type of the **data1** file.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data1

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

**data1** is a **gzip** compressed file, therefore rename it properly and then decompress by using **gzip** command.

bandit12@bandit:/tmp/bandit12\_ahmer$ mv data1 data2.tar.gz

bandit12@bandit:/tmp/bandit12\_ahmer$ gzip -d data2.tar.gz

After decompression, we got **data2.tar** file. Again check its file type.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data2.tar

data2.tar: bzip2 compressed data, block size = 900k

**data2.tar** is a **bzip2** compressed file, therefore rename it properly and then decompress by using **bzip2** command.

bandit12@bandit:/tmp/bandit12\_ahmer$ mv data2.tar data3.tar.bz2

bandit12@bandit:/tmp/bandit12\_ahmer$ bzip2 -d data3.tar.bz2

We got **data3.tar** file after decompression. Check its file type again.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data3.tar

data3.tar: gzip compressed data, was "data4.bin", last modified: Tue Oct 16 12:00:23 2018, max compression, from Unix

**data3.tar** is a **gzip** compressed file, therefore rename and extract it.

bandit12@bandit:/tmp/bandit12\_ahmer$ mv data3.tar data4.tar.gz

bandit12@bandit:/tmp/bandit12\_ahmer$ gzip -d data4.tar.gz

We have **data4.tar** file now. Check its file type.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data4.tar

data4.tar: POSIX tar archive (GNU)

**data4.tar** is a **tar** archive, therefore, we can extract it using **tar** command.

bandit12@bandit:/tmp/bandit12\_ahmer$ tar xf data4.tar

We get **data5.bin** from tar archive. Check its file type again.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data5.bin

data5.bin: POSIX tar archive (GNU)

**data4.bin** is a **tar** archive. Therefore, rename and extract it.

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

bandit12@bandit:/tmp/bandit12\_ahmer$ tar xf data5.tar

We get **data6.bin** from tar archive. Check it file type again.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data6.bin

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

This time it is a **bzip2** compressed file. Rename and extract it accordingly.

bandit12@bandit:/tmp/bandit12\_ahmer$ mv data6.bin data6.tar.bz2

bandit12@bandit:/tmp/bandit12\_ahmer$ bzip2 -d data6.tar.bz2

We get **data6.tar** from **bzip2** archive. Check its content type.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data6.tar

data6.tar: POSIX tar archive (GNU)

**data6.tar** is a tar archive, therefore extract it using **tar** command.

bandit12@bandit:/tmp/bandit12\_ahmer$ tar xf data6.tar

This time we have **data8.bin** from tar archive. Check its file type.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data8.bin

data8.bin: gzip compressed data, was "data9.bin", last modified: Tue Oct 16 12:00:23 2018, max compression, from Unix

**data8.bin** is a **gzip** compressed file. Therefore, rename and extract it as follows.

bandit12@bandit:/tmp/bandit12\_ahmer$ mv data8.bin data8.tar.gz

bandit12@bandit:/tmp/bandit12\_ahmer$ gzip -d data8.tar.gz

We got **data8.tar** from **gzip** archive. Check its file type.

bandit12@bandit:/tmp/bandit12\_ahmer$ file data8.tar

data8.tar: ASCII text

Finally, we get a **text** file. Print its contents now.

bandit12@bandit:/tmp/bandit12\_ahmer$ cat data8.tar

The password is 8ZjyCRiBWFYkneahHwxCv3wb2a1ORpYL

We have found the password of **bandit13** user.

**Level 13 - Level 14:**

**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

**Solution:**

This time you will get a **private SSH key**, that can be used to login as **bandit14** user without any password.

bandit13@bandit:~$ ls

sshkey.private

Use **ssh** command to login as **bandit14** user by using this **private SSH key**.

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

After successful login, password of **bandit14** user is now accessible. Obtain it from the file **/etc/bandit\_pass/bandit14**.

bandit14@bandit:~$ cat /etc/bandit\_pass/bandit14

4wcYUJFw0k0XLShlDzztnTBHiqxU3b3e

**Level 14 - Level 15:**

**Goal:**

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

**Solution:**

To obtain the password of **bandit15**, we need to send the password of **bandit14** at port 30000.

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

Correct!

BfMYroe26WYalil77FoDi9qh59eK5xNr

**Level 15 - Level 16:**

**Goal:**

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…

**Solution:**

To obtain password of **bandit16** user, we need to send password of **bandit15** user at **port 30001** in encrypted form.

bandit15@bandit:~$ echo "BfMYroe26WYalil77FoDi9qh59eK5xNr" | openssl s\_client -connect localhost:30001 -ign\_eof

CONNECTED(00000003)

---

Correct!

cluFn7wTiGryunymYOu4RcffSxQluehd

closed

We have received the password of **bandit16** user.

**Level 16 - Level 17:**

**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.

**Solution:**

Find the open ports between range **31000-32000**.

bandit16@bandit:~$ nmap localhost -p 31000-32000

Starting Nmap 7.40 ( https://nmap.org ) at 2020-03-25 08:37 CET

Nmap scan report for localhost (127.0.0.1)

Host is up (0.00027s latency).

Not shown: 999 closed ports

PORT STATE SERVICE

31518/tcp filtered unknown

31790/tcp open unknown

Nmap done: 1 IP address (1 host up) scanned in 1.24 seconds

**31790/tcp** is our target port. Therefore, send password of **bandit16** user at port **31790**.

bandit16@bandit:~$ echo "cluFn7wTiGryunymYOu4RcffSxQluehd" | openssl s\_client -connect localhost:31790 -ign\_eof

CONNECTED(00000003)

---

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-----

closed

This time we don't get the password of **bandit17** user but instead we have the **private SSH key** to login without password.

We need to save this **private SSH key** in a text file. Therefore, create a temporary directory.

bandit16@bandit:~$ mkdir /tmp/bandit16\_ahmer

bandit16@bandit:~$ cd /tmp/bandit16\_ahmer

Create a file using **nano** editor and copy/paste the **private SSH key** therein.

bandit16@bandit:~$ nano /tmp/bandit16\_ahmer/sshkey.txt

Change permissions of this file.

bandit16@bandit:~$ chmod 600 /tmp/bandit16\_ahmer/sshkey.txt

Now, we can login by using this SSH key as follows.

bandit16@bandit:~$ ssh bandit17@localhost -i /tmp/bandit16\_ahmer/sshkey.txt

We have been successfully login as **bandit17** user. We can now get the password from **/etc/bandit\_pass** directory.

bandit17@bandit:~$ cat /etc/bandit\_pass/bandit17

xLYVMN9WE5zQ5vHacb0sZEVqbrp7nBTn

**Level 17 - Level 18:**

**Goal:**

There are 2 files in the **home** directory: **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**

**Solution:**

Check the difference in **password.new** and **password.old** files.

bandit17@bandit:~$ diff passwords.old passwords.new

42c42

< hlbSBPAWJmL6WFDb06gpTx1pPButblOA

---

> kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd

We can now login as **bandit18** user using password "kfBf3eYk5BPBRzwjqutbbfE887SVc5Yd".

**Level 18 - Level 19:**

**Goal:**

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

**Solution:**

When we try to login as **bandit18** user, the session ends immediately.

bandit17@bandit:~$ ssh bandit18@localhost

...

Byebye !

Connection to localhost closed.

Login as **bandit18** user and bypass the pseudo session allocation.

bandit16@bandit:~$ ssh bandit18@localhost -T

We have been successfully login as **bandit16** user and bypass the **.bashrc** file.

Check **readme** file and obtain the password of **bandit19** user.

cat readme

IueksS7Ubh8G3DCwVzrTd8rAVOwq3M5x

**Level 19 - Level 20:**

**Goal:**

To gain access to the next level, you should use the **setuid** binary in the **home** directory. 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.

**Solution:**

In this question, you have been provided a **setuid** binary command that executes commands as **bandit20** user.

To check the usage, execute it without any argument.

bandit19@bandit:~$ ./bandit20-do

Run a command as another user.

Example: ./bandit20-do id

We know the location of the password therefore pass the command to display its contents to **bandit20-do** script.

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

GbKksEFF4yrVs6il55v6gwY5aVje5f0j

We have found the password of **bandit20** user.

**Level 20 - Level 21:**

**Goal:**

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 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

**Solution:**

We have a **setuid** binary command in this question. Execute this command without any argument.

bandit20@bandit:~$ ./suconnect

Usage: ./suconnect <portnumber>

As given in the challenge, 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.

Start a **netcat** listener on port **55555** that will return the **bandit20** password to connecting program.

bandit20@bandit:~$ echo GbKksEFF4yrVs6il55v6gwY5aVje5f0j | nc -l localhost -p 55555 &

[1] 25315

Now execute the **suconnect** script to connect with port **55555**.

bandit20@bandit:~$ ./suconnect 55555

Read: GbKksEFF4yrVs6il55v6gwY5aVje5f0j

Password matches, sending next password

gE269g2h3mw3pwgrj0Ha9Uoqen1c9DGr

We have got the password of **bandit21** user.

**Level 21 - Level 22:**

**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.

**Solution:**

Let's have a look, what is available in **/etc/cron.d** directory.

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

atop cronjob\_bandit22 cronjob\_bandit23 cronjob\_bandit24

Our core interest is **bandit22** user this time, therefore check what is running by the **cronjob\_bandit22** schedule.

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

Execute **cronjob\_bandit22.sh** script and observe the output.

bandit21@bandit:~$ /usr/bin/cronjob\_bandit22.sh

chmod: changing permissions of '/tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv': Operation not permitted

/usr/bin/cronjob\_bandit22.sh: line 3: /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv: Permission denied

Check the contents of **/tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv** file.

bandit21@bandit:~$ cat /tmp/t7O6lds9S0RqQh9aMcz6ShpAoZKF7fgv

Yk7owGAcWjwMVRwrTesJEwB7WVOiILLI

We have found the password of **bandit22** user.

**Level 22 - Level 23:**

**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.

**Solution:**

Just like previous challenge, there is also a **cron** job. Lets have a look into **/etc/cron.d** directory.

bandit22@bandit:~$ ls /etc/cron.d

atop cronjob\_bandit22 cronjob\_bandit23 cronjob\_bandit24

Our concern is the password of **bandit23** user, therefore **cronjob\_bandit23** might be useful for us.

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

We have located the script, let's have a look into its code.

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

This script is copying the passwordfile of the login user to a temporary file in **/tmp** directory. The filename of temporary file is generated by using the username of the current user.

We are executing the following code segment from this script after replacing the **$mytarget** variable with a literal **bandit23**. This way we will obtain the name of the file that contains the password of **bandit23** user.

bandit22@bandit:/usr/bin$ echo I am user bandit23 | md5sum | cut -d ' ' -f 1

8ca319486bfbbc3663ea0fbe81326349

Check the contents of this temporary file.

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

jc1udXuA1tiHqjIsL8yaapX5XIAI6i0n

We have obtained the password of **bandit23** user.

**Level 23 - Level 24:**

**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: 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…

**Solution:**

Check **/etc/cron.d** directory.

bandit23@bandit:~$ ls /etc/cron.d

atop cronjob\_bandit22 cronjob\_bandit23 cronjob\_bandit24

Our concern is **bandit24** password, therefore check the **cronjob\_bandit24** cronjob.

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

Now, we got the location of the script that is being executed by above cron job.

Check the code of this script.

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

By analyzing this script, we find out that it is executing all scripts from **/var/spool/bandit24** directory.

By listing the permissions of the **/var/spool** directory, we have observed that we have write permissions on **/var/spool/bandit24** directory.

bandit23@bandit:~$ ls -al /var/spool

total 1348

drwxr-xr-x 5 root root 4096 Oct 16 2018 .

drwxr-xr-x 11 root root 4096 Oct 16 2018 ..

drwxrwx-wx 35 root bandit24 1359872 Mar 27 09:04 bandit24

drwxr-xr-x 3 root root 4096 Oct 16 2018 cron

lrwxrwxrwx 1 root root 7 Oct 16 2018 mail -> ../mail

drwx------ 2 root root 4096 Jan 14 2018 rsyslog

Let's create a custom script of our own in this directory. The target is to obtain the password file of **bandit24** from **/etc/bandit\_pass/** to **/tmp** directory.

bandit23@bandit:~$ nano /var/spool/bandit24/myscript.sh

and following lines of code.

#!/bin/bash

cat /etc/bandit\_pass/bandit24 > /tmp/ahmer\_bandit24

Make this file executable.

bandit23@bandit:~$ chmod 777 /var/spool/bandit24/myscript.sh

Now wait for a few seconds and then check the file **/tmp/ahmer\_bandit24**.

bandit23@bandit:~$ cat /tmp/ahmer\_bandit24

UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ

We have got the password of **bandit24** user.

**Level 24 - Level 25:**

**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**.

**Solution:**

We have the password of **bandit24** user, but the 4 digit pin is unknown. However, we can use brute force technique to generate all possible combinations of 4 digit and then send each of them to the daemon.

The possible number of combinations for **n** digits is **10n** . Therefore, in present scenario it will be

**104 = 10000**

We are generating a space separated list of bandit24 password with all the possible combinations of digits between 0000 and 9999 by using the following command.

bandit24@bandit:~$ for i in $(seq -f "%04g" 0 9999); do echo "UoMYTrfrBFHyQXmg6gzctqAwOmw1IohZ" $i >> /tmp/ahmer\_pins; done;

Now, we have a text file that contains all the possible combinations. we can submit this file to the daemon running at port 30002 to obtain the password of **bandit25** user.

bandit24@bandit:~$ cat /tmp/ahmer\_pins | nc localhost 30002 | grep password

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.

The password of user bandit25 is uNG9O58gUE7snukf3bvZ0rxhtnjzSGzG

**Level 25 - Level 26:**

**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.

**Solution:**

We have a **private ssh key** file located in **bandit25** user’s **home** directory.

bandit25@bandit:~$ ls

bandit26.sshkey

Login as **bandit26** user by using this **private ssh key** file.

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

...

Connection to localhost closed.

When we attempt to login as **bandit26** user. The session will end immediately.

Login as **bandit25** user and check the default shell of **bandit26** user.

bandit25@bandit:~$ grep bandit26 /etc/passwd

bandit26:x:11026:11026:bandit level 26:/home/bandit26:/usr/bin/showtext

Check the contents of the custom shell of **bandit26** user.

bandit25@bandit:~$ cat /usr/bin/showtext

#!/bin/sh

export TERM=linux

more ~/text.txt

exit 0

Above script invokes a **more** command and thereafter **exit** command that causes the shell to terminate and close the connection.

Actually the size of our **PuTTY** command window is large enough for the more command to display 100% of the file contents. Therefore, **more** command executes successfully and then exit command executed and closed the connection.

Reduce the height of the **PuTTY** window and then try to login again.

[](https://2.bp.blogspot.com/-7YnrE9qnEaE/XoQ_naSOgKI/AAAAAAAAIBs/Uz79stJeBEE_O4suJbvNmAfYAXBUFYftwCLcBGAsYHQ/s1600/01-bandit-wargame-walkthrough-level-25-more.png)

The **more** command is in control now.

Enable **vi** editor by pressing **'v'**at the **more** window.

[](https://4.bp.blogspot.com/-czVU5OeLDUA/XoQ_nZofVRI/AAAAAAAAIB0/VjQik3nDWzIzneuDIa_R7_DzSccL5tH3gCLcBGAsYHQ/s1600/02-bandit-wargame-walkthrough-level-25-vi.png)

Now leverage the functionality of the **vi** editor to invoke a custom shell.

[](https://3.bp.blogspot.com/-V7x2FS9FrMo/XoQ_nCS-vYI/AAAAAAAAIBw/oIPI06tyys8aOu9wbdyXceMBZ_ocA64hQCLcBGAsYHQ/s1600/03-bandit-wargame-walkthrough-level-25-shell.png)

Now execute **sh** command at **vi** terminal to spawn a shell.

[](https://2.bp.blogspot.com/-vBVZqcED_e8/XoQ_n86HE7I/AAAAAAAAIB4/kYglPo1I1PMGXDHs1AknSB89iDbdZFaZwCLcBGAsYHQ/s1600/04-bandit-wargame-walkthrough-level-25-sh.png)

We are successfully login as the **bandit26** user.

**Level 26 - Level 27:**

**Goal:**

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

**Solution:**

Not much hint given in this challenge. But we have a **suid** executable in **home** directory.

bandit26@bandit:~$ ls

bandit27-do text.txt

Try to execute this file.

bandit26@bandit:~$ ./bandit27-do

Run a command as another user.

Example: ./bandit27-do id

We know that the passwords are located in **/etc/bandit\_pass/** directory.

Display the content of **bandit27** passwordfile.

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

3ba3118a22e93127a4ed485be72ef5ea

We have got the password of **bandit27** user.

**Level 27 - Level 28:**

**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.

**Solution:**

Create a temporary directory to clone the given **git** repository. (refer to our previous post on [git](https://www.centlinux.com/2018/12/local-git-server-and-client-in-rhel-7.html) command)

bandit27@bandit:~$ mkdir /tmp/bandit27\_ahmer

bandit27@bandit:~$ cd /tmp/bandit27\_ahmer

Clone the given **git** repository as follows.

bandit27@bandit:/tmp/bandit27\_ahmer$ 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.

Now check the contents of this **git** repository.

bandit27@bandit:/tmp/bandit27\_ahmer$ cd repo

bandit27@bandit:/tmp/bandit27\_ahmer/repo$ ls

README

Display the content of **README** file.

bandit27@bandit:/tmp/bandit27\_ahmer/repo$ cat README

The password to the next level is: 0ef186ac70e04ea33b4c1853d2526fa2

**Level 28 - Level 29:**

**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.

**Solution:**

Create a directory in **/tmp** directory to clone the git repository.

bandit28@bandit:~$ mkdir /tmp/bandit28\_ahmer

bandit28@bandit:~$ cd /tmp/bandit28\_ahmer

Clone the given **git** repository.

bandit28@bandit:/tmp/bandit28\_ahmer$ 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:

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.

Check the contents of this **git** repository.

bandit28@bandit:/tmp/bandit28\_ahmer$ cd repo

bandit28@bandit:/tmp/bandit28\_ahmer/repo$ ls

README.md

bandit28@bandit:/tmp/bandit28\_ahmer/repo$ cat README.md

# Bandit Notes

Some notes for level29 of bandit.

## credentials

- username: bandit29

- password: xxxxxxxxxx

The password is not available here. But remember that we are working on a [version control system](https://en.wikipedia.org/wiki/Version_control). Therefore, check the log of the changes in this repository.

bandit28@bandit:/tmp/bandit28\_ahmer/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

**'fix info leak'** commit looks suspicious to me. Therefore, we are checking the changes between consecutive commits.

bandit28@bandit:/tmp/bandit28\_ahmer/repo$ git diff 073c27c130e6ee407e12faad1dd3848a110c4f95 186a1038cc54d1358d42d468cdc8e3cc28a93fcb

diff --git a/README.md b/README.md

index 5c6457b..3f7cee8 100644

--- a/README.md

+++ b/README.md

@@ -4,5 +4,5 @@ Some notes for level29 of bandit.

## credentials

- username: bandit29

-- password: xxxxxxxxxx

+- password: bbc96594b4e001778eee9975372716b2

We have got the password of the **bandit29** user.

**Level 29 - Level 30:**

**Goal:**

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.

**Solution:**

Create a temporary directory to clone given **git** repository.

bandit29@bandit:~$ mkdir /tmp/bandit29\_ahmer

bandit29@bandit:~$ cd /tmp/bandit29\_ahmer

Clone the given **git** repository.

bandit29@bandit:/tmp/bandit29\_ahmer$ git clone ssh://bandit29-git@localhost/home/bandit29-git/repo

Cloning into 'repo'...

Could not create directory '/home/bandit29/.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/bandit29/.ssh/known\_hosts).

This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit29-git@localhost's 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.

Check the contents of this repository.

bandit29@bandit:/tmp/bandit29\_ahmer$ cd repo

bandit29@bandit:/tmp/bandit29\_ahmer/repo$ ls

README.md

Print contents of **README.md** file.

bandit29@bandit:/tmp/bandit29\_ahmer/repo$ cat README.md

# Bandit Notes

Some notes for bandit30 of bandit.

## credentials

- username: bandit30

- password: <no passwords in production!>

A clear hint is given in the file that, the password is not available in **production** branch, therefore, it may be available on **dev** branch.

List all the branches of this **git** repository.

bandit29@bandit:/tmp/bandit29\_ahmer/repo$ git branch -a

\* master

remotes/origin/HEAD -> origin/master

remotes/origin/dev

remotes/origin/master

remotes/origin/sploits-dev

We have a **dev** branch available here. Switch to **dev** branch to see its contents.

bandit29@bandit:/tmp/bandit29\_ahmer/repo$ git checkout dev

Branch dev set up to track remote branch dev from origin.

Switched to a new branch 'dev'

Print the contents of **README.md** file now.

bandit29@bandit:/tmp/bandit29\_ahmer/repo$ cat README.md

# Bandit Notes

Some notes for bandit30 of bandit.

## credentials

- username: bandit30

- password: 5b90576bedb2cc04c86a9e924ce42faf

We have found the password of **bandit30** user.

**Level 30 - Level 31:**

**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.

**Solution:**

Create a temporary directory to clone the **git** repository.

bandit30@bandit:~$ mkdir /tmp/bandit31\_ahmer

bandit30@bandit:~$ cd /tmp/bandit31\_ahmer

Clone **git** repository as follows.

bandit30@bandit:/tmp/bandit31\_ahmer$ git clone ssh://bandit30-git@localhost/home/bandit30-git/repo

Cloning into 'repo'...

Could not create directory '/home/bandit30/.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/bandit30/.ssh/known\_hosts).

This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit30-git@localhost's password:

remote: Counting objects: 4, done.

remote: Total 4 (delta 0), reused 0 (delta 0)

Receiving objects: 100% (4/4), done.

Check the contents of this **git** repository.

bandit30@bandit:/tmp/bandit31\_ahmer$ cd repo

bandit30@bandit:/tmp/bandit31\_ahmer/repo$ ls

README.md

Print contents of **README.md** file.

bandit30@bandit:/tmp/bandit31\_ahmer/repo$ cat README.md

just an epmty file... muahaha

This time we got a file without any change log history. Therefore, lets check the other objects related to this repository.

Check if there are any tags in this repository.

bandit30@bandit:/tmp/bandit31\_ahmer/repo$ git tag

secret

We have got a suspicious tag here. Check its contents now.

bandit30@bandit:/tmp/bandit31\_ahmer/repo$ git show secret

47e603bb428404d265f59c42920d81e5

We have the password of **bandit31** user.

**Level 31 - Level 32:**

**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.

**Solution:**

Create a directory in **/tmp** to clone the given **git** repository.

bandit31@bandit:~$ mkdir /tmp/bandit32\_ahmer

bandit31@bandit:~$ cd /tmp/bandit32\_ahmer

Clone the **git** repository.

bandit31@bandit:/tmp/bandit32\_ahmer$ git clone ssh://bandit31-git@localhost/home/bandit31-git/repo

Cloning into 'repo'...

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:

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.

Check contents of **git** repository.

bandit31@bandit:/tmp/bandit32\_ahmer$ cd repo

bandit31@bandit:/tmp/bandit32\_ahmer/repo$ ls

README.md

Print contents of **README.md** file.

bandit31@bandit:/tmp/bandit32\_ahmer/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

A clear hint is given in this file that, this time we have to create and push a file in **git** repository.

Create the required file as follows.

bandit31@bandit:/tmp/bandit32\_ahmer/repo$ echo "May I come in?" > key.txt

Add **key.txt** file to **git** repository.

bandit31@bandit:/tmp/bandit32\_ahmer/repo$ git add key.txt -f

Commit changes to local **git** repository.

bandit31@bandit:/tmp/bandit32\_ahmer/repo$ git commit

Push the changes to the repository origin.

bandit31@bandit:/tmp/bandit32\_ahmer/repo$ git push origin

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 4 threads.

Compressing objects: 100% (2/2), done.

Writing objects: 100% (3/3), 325 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'

We have obtain the password of **bandit32** user.

**Level 32 - Level 33:**

**Goal:**

After all this **git** stuff its time for another **escape**. Good luck!

**Solution:**

After login as **bandit33**, we get the following welcome message.

WELCOME TO THE UPPERCASE SHELL

We have checked that, no command either in lowercase or uppercase are not working in this shell. It looks like that the shell is converting each command to uppercase before executing, therefore, no command is being able to execute.

It means that, this shell is of no use for us. So we try to invoke a regular shell (such as **sh** or **bash**).

We can spawn a shell using a shortcut command **'$0'**. This is in fact an environment variable that holds the shell command that is used to execute all the commands in current shell.

Execute**'$0'** to invoke a regular shell.

>> $0

$

We don't have any hint to find the **bandit33** password. But we know that the passwords are located in **/etc/bandit\_pass** directory.

Display the content of **bandit33** passwordfile.

$ cat /etc/bandit\_pass/bandit33

c9c3199ddf4121b10cf581a98d51caee

**Level 33 - Level 34:**

**Goal:**

At this moment, level 34 does not exist yet.

**Solution:**

Check contents of **home** directory.

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!