ESCUELA COLOMBIANA DE INGENIERIA
JULIO GARAVITO

IT SECURITY AND PRIVACY
GRUPO 1L

LINUX PRACTICE
LAB 2

SUBMITTED BY:
JUAN PABLO FERNANDEZ GONZALES
MARIA VALENTINA TORRES MONSALVE

SUBMITTED TO:
Ing. DANIEL ESTEBAN VELA LOPEZ

BOGOTÁ D.C.

DATE:
03/02/2025

Table of Content

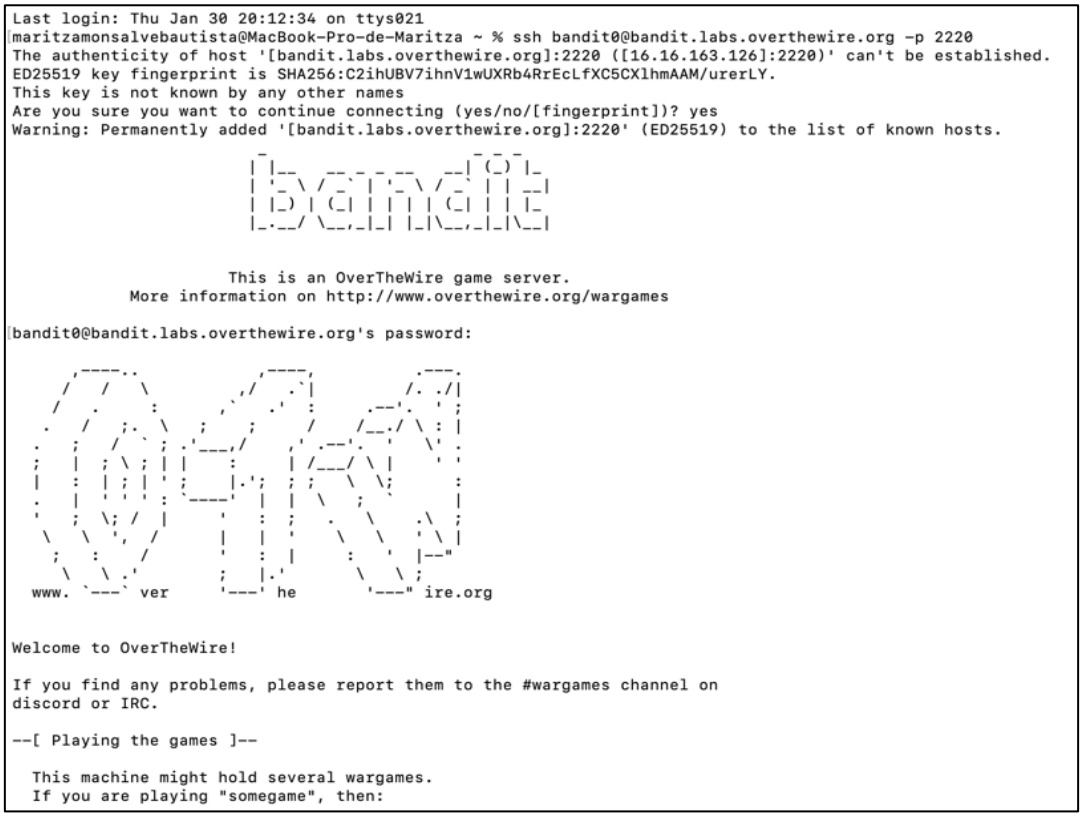
Level 0

Objective

In this exercise the objective is to log in to the game using SSH, we must enter with the user bandit0 and the password bandit0 and the host bandit.labs.overthewire.org through port 2220.

Methodology

The first thing we will do is enter the computer terminal, since we will be using it to interact with the Bandit server. The command used to establish the connection to the Bandit server is *'ssh bandit0@bandit.labs.overthewire.org -p 2220'*



- **ssh:** Secure Shell, is a network protocol with which we can establish a remote connection from the command line. This connection is secure since the data that is sent cannot be seen by third parties since all traffic sent and received is encrypted.
- **bandit0@bandit.labs.overthewire.org:** This part of the command tells us which server is going to make the ssh connection to, the first part ***bandit0*** is the user, **@** separates the user and the host, the remaining ***bandit.labs.overthewire.org*** is the name of the server host for the Bandit challenge.
- **-p 2220: -p** is the tag that is used to specify the port of the server with which you want to establish the connection since by default the SSH protocol connects on 22, after this label we enter the port number that to be able to start the challenge will be **2220.**

After executing the command, it asks us if we are sure to establish the connection, after accepting we are asked for the bandit0 password **.** We will finally have full access to the server.

```
 For your convenience we have installed a few useful tools which you can find
 in the following locations:

    * gef (https://github.com/hugsy/gef) in /opt/gef/
    * pwndbg (https://github.com/pwndbg/pwndbg) in /opt/pwndbg/
    * gdbinit (https://github.com/gdbinit/Gdbinit) in /opt/gdbinit/
    * pwntools (https://github.com/Gallopsled/pwntools)
    * radare2 (http://www.radare.org/)

--[ More information ]--

  For more information regarding individual wargames, visit
  http://www.overthewire.org/wargames/

  For support, questions or comments, contact us on discord or IRC.

  Enjoy your stay!

bandit0@bandit:~$
```

Results: ZjLjTmM6FvvyRnrb2rfNWOZOTa6ip5If

After entering the command and password was accepted. The user will already have access to the Bandit server with the user bandit0, they will give us some tips to navigate and perform actions on it.

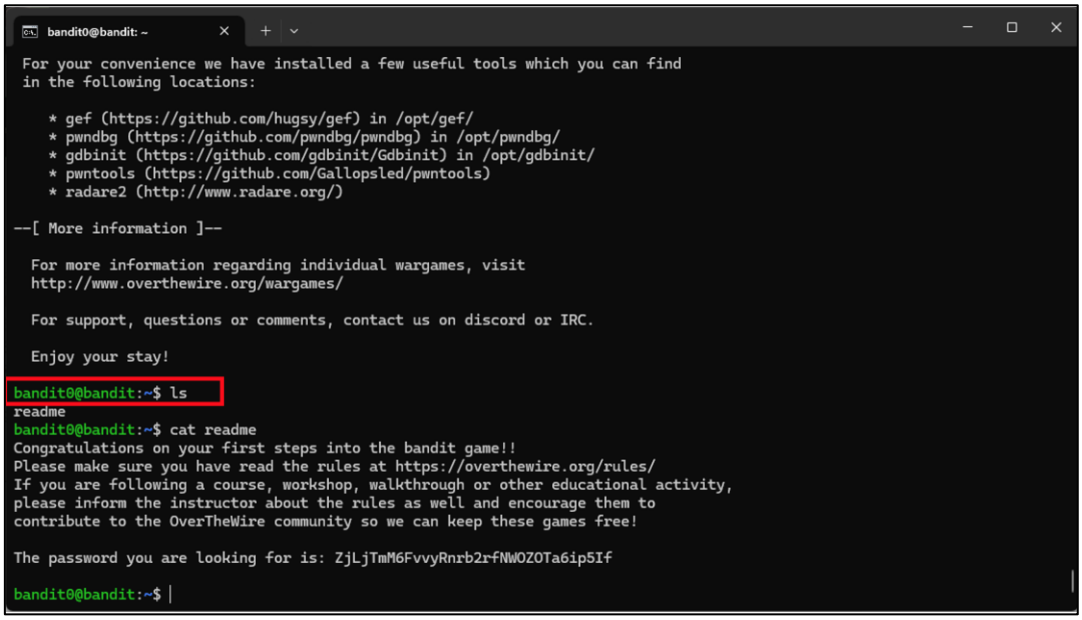
Level 0 → Level 1

Objective

The objective of this level will be to be able to access the file called ***'readme'*** located in the home directory and obtain the password for the next level that as in the previous level we will connect by ssh on port 2220 with the user bandit1, so it is very important to get the password to be able to progress in the game.

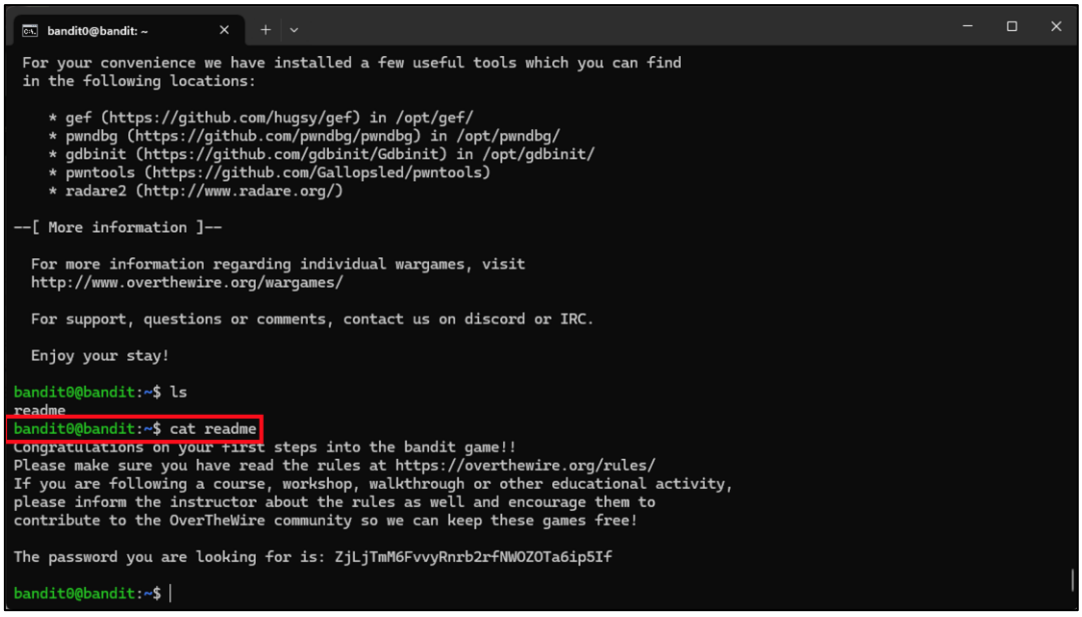
Methodology

The first thing we'll do is we'll list the contents of the directory we're located in to find the ***readme*** file using '***ls***'



- **ls:** This command allows us to list all the files or directories in the current directory by default.

After having found the ***readme file,*** the next step will be to read the contents of it in order to obtain the password, for this we will use the ***command 'cat readme'***



- **Cat Readme:** Cat is a command that allows us to read the contents of files and issue them on the command line, its basic structure is 'cat <file_name>'

Results: 263JGJPfgU6LtdEvgfWU1XP5yac29mFx

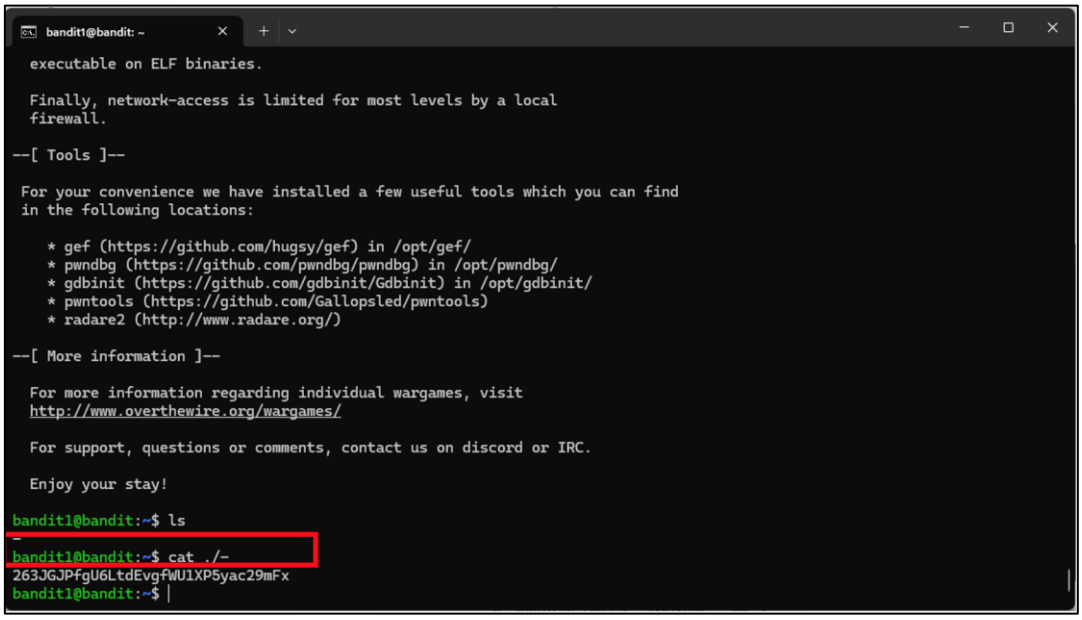
Level 1 → Level 2

Objective

The objective of this level will be to be able to access the file called *'-'* located in the home directory and obtain the password for the next level to which we will connect by ssh on port 2220 with the user bandit2.

Methodology

After having found the ***-file,*** the next step will be to read the contents of it in order to obtain the password, for this we will use the ***command 'cat ./-'***



- **cat ./-:** 'cat' is a command that allows us to read the contents of files and display them in the command line. In this case, we use ***"cat ./<file_name>"*** because '**./**' specifies the current directory. Without it, entering ***"cat -"*** would be interpreted as an option rather than a filename. If we enter this command without specifying a file, it will prompt for user input and print whatever we type to the console instead of displaying the file's contents.

Results: MNk8KNH3Usiio41PRUEoDFPqfxLPlSmx

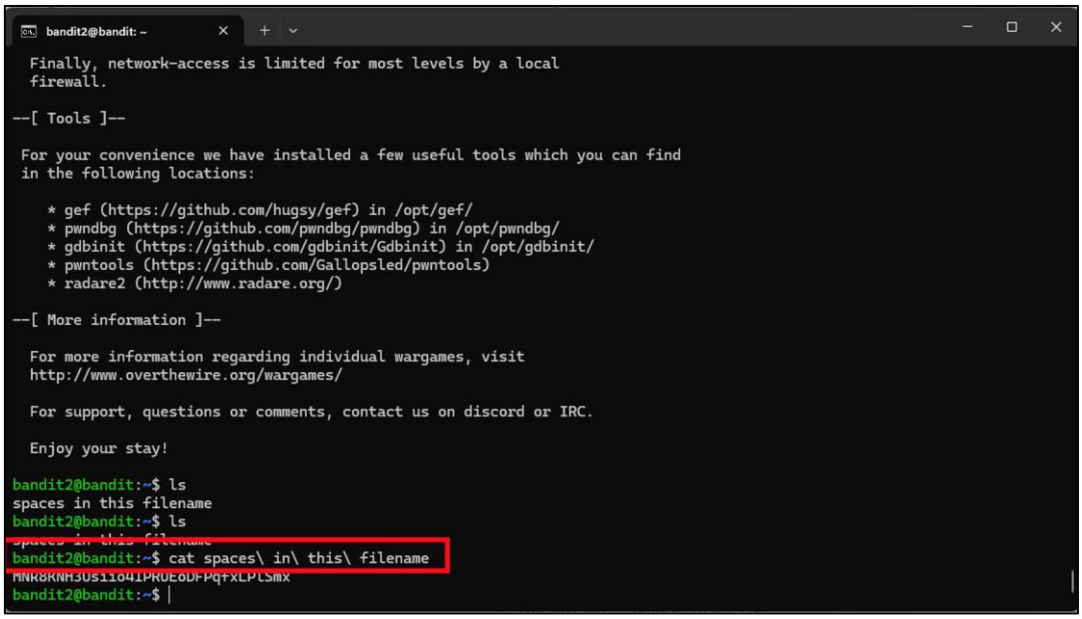
Level 2 → Level 3

Objective

The objective of this level will be to be able to access the file called ***'spaces in this filename'*** located in the home directory and obtain the password for the next level to which we will connect by ssh on port 2220 with the user bandit3.

Methodology

After having found the ***-file,*** the next step will be to read the contents of it in order to obtain the password, for this we will use the ***command 'cat spaces\ in\ this\ filename'***



- **cat *spaces\ in\ this\ filename* :** 'cat' is a command that allows us to read the contents of files and display them in the command line. In this case, we use "**spaces\ in\ this\ filename"** because **"\"** se utiliza para leer y mostrar el contenido de un archivo cuyo nombre contiene espacios. En la línea de comandos, los espacios se interpretan como separadores de argumentos, por lo que es necesario escaparlos con una barra invertida, comillas simples o dobles. Por ejemplo, "**spaces in this filename"** o **'spaces in this filename'.**

Results: 2WmrDFRmJIq3IPxneAaMGhap0pFhF3NJ

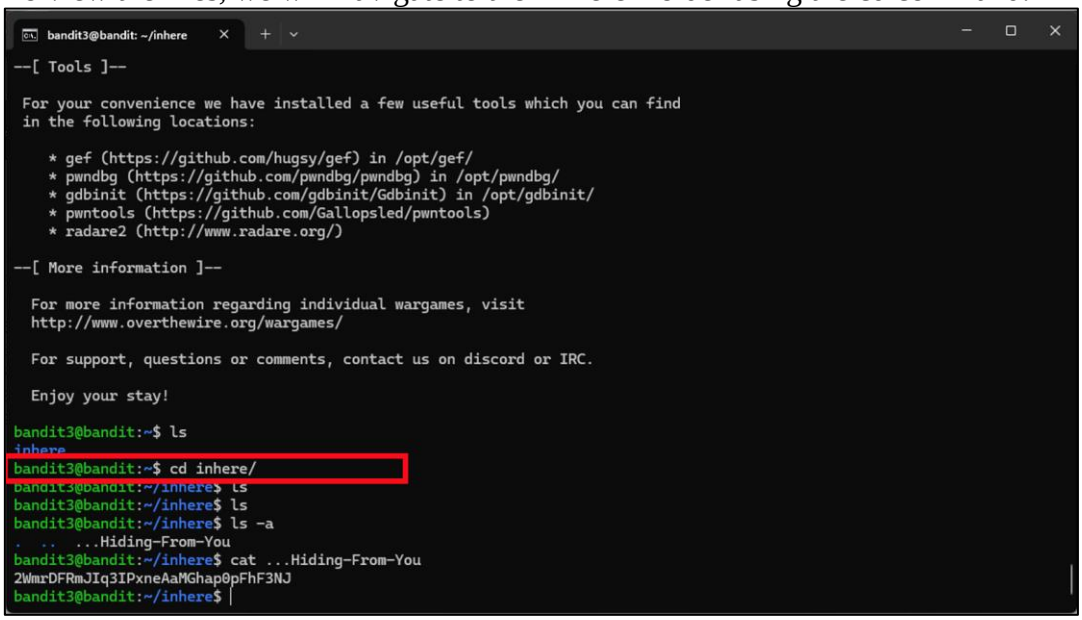
Level 3 → Level 4

Objective

The objective of this level will be to be able to access a hidden file located in the **inhere directory** and obtain the password for the next level to which we will connect by ssh on port 2220 with the user bandit4.

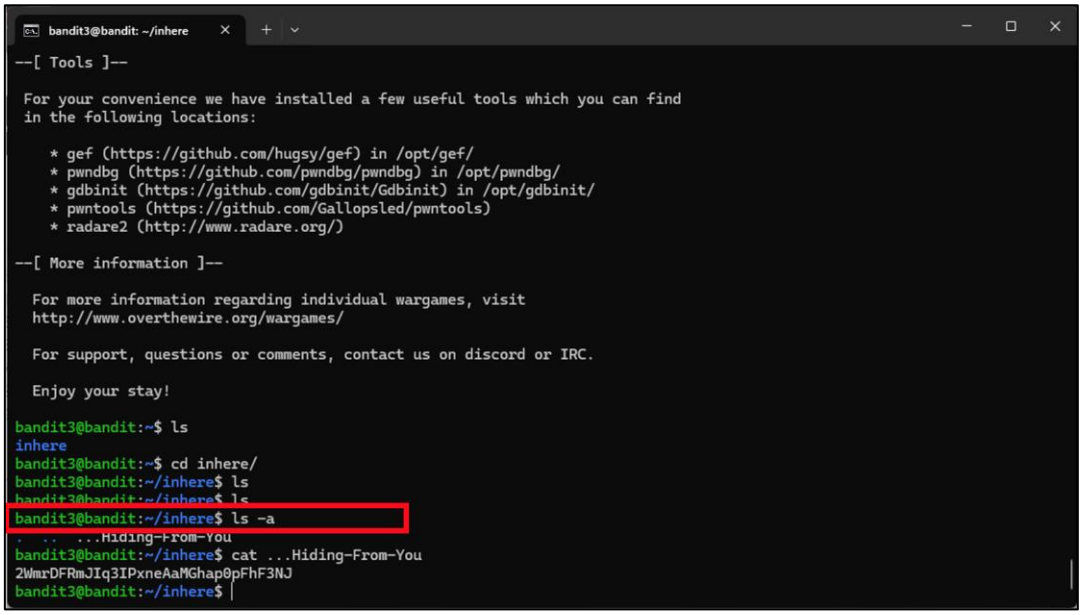
Methodology

To view the files, we will navigate to the "inhere" folder using the cd command.



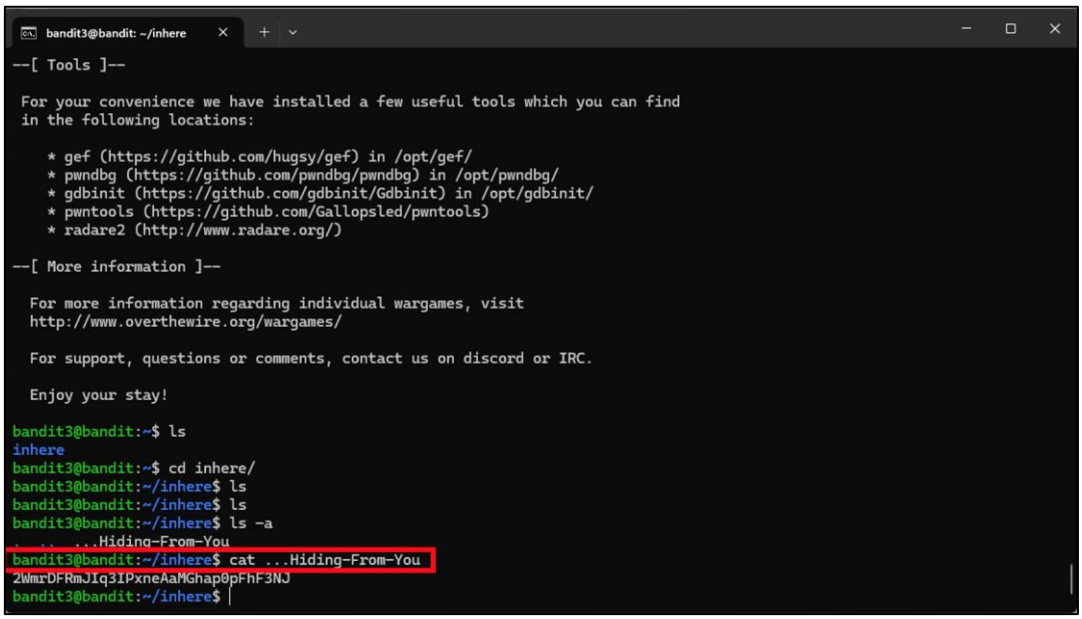
- **cd inhere/:** With the **'cd'** command, we can navigate between different directories in the system. In this case, we entered the name of the directory we wanted to go to.

The first thing we'll do is we'll list the contents of the directory we're located in to find the file using '***ls -a*** '



- **ls -a:** This command allows us to list all the files or directories in the current directory by default, including hidden files thanks to the **-a** flag.

After having found the file**,** the next step will be to read the contents of it in order to obtain the password, for this we will use the command ***'cat …Hiding-From-You'***



Results: 4oQYVPkxZOOEOO5pTW81FB8j8lxXGUQw

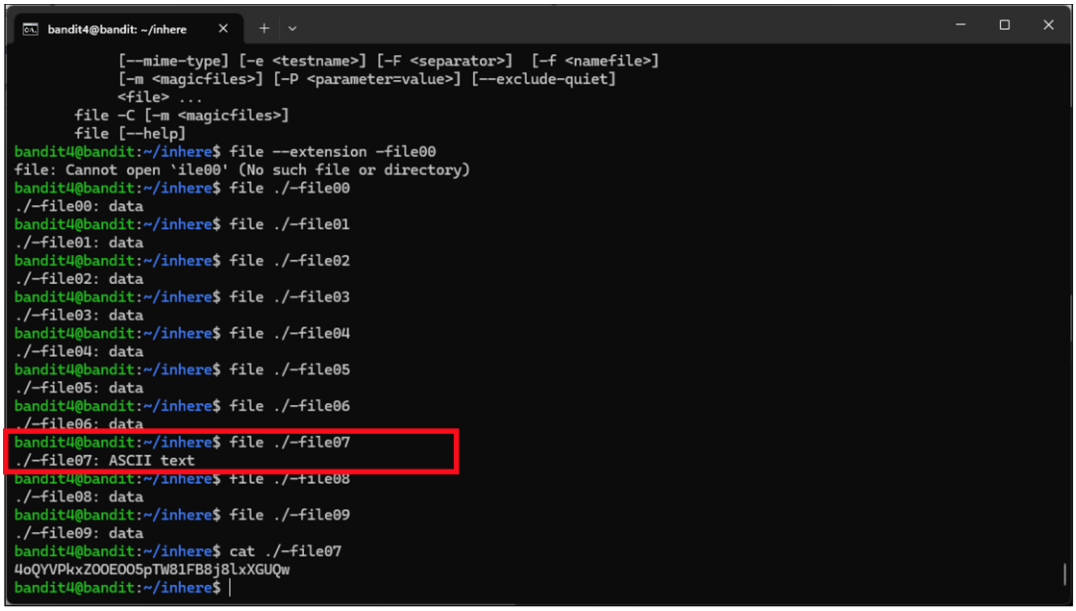
Level 4 → Level 5

Objective

The objective of this level is to find, among the multiple files in the **inhere** directory, the only one that is human-readable and obtain the password for the next level, to which we will connect by SSH on port 2220 with the user **bandit5**.

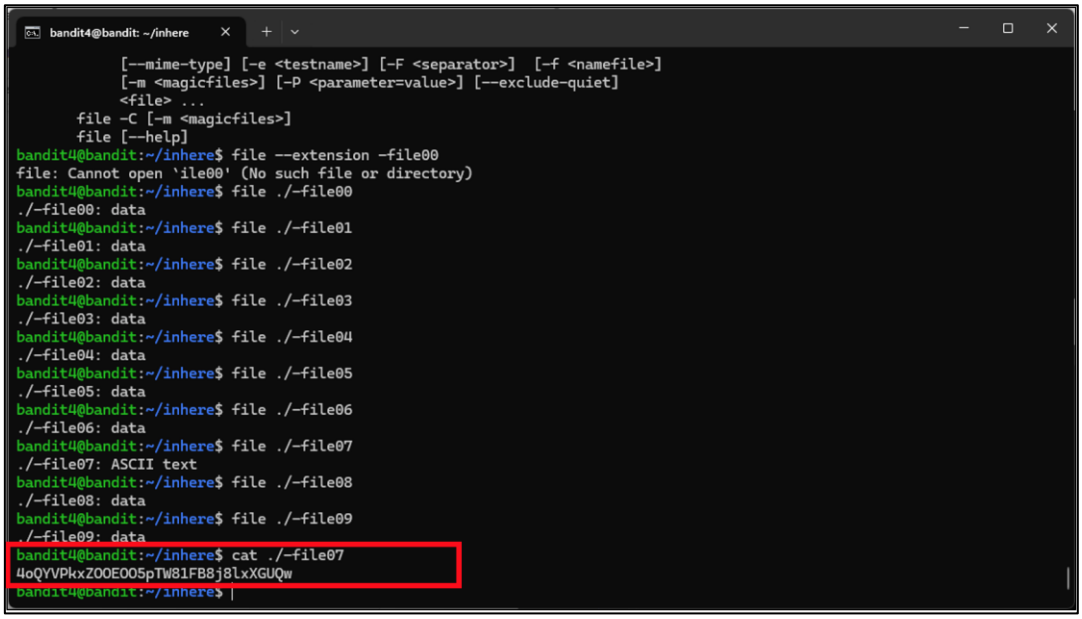
Methodology

Once inside the **inhere** directory, we will look for a file with an extension that we can read and obtain the password, using the command **file ./<filename>**.



- **file ./<filename>**: The **file** command analyzes a file and displays its type, such as text, binary, image, or executable. The **./** prefix specifies that the file is in the current directory.

After identifying the file that is human-readable, we use the ***cat*** command. Just like with ***file***, we use ***./*** to prevent - from being interpreted as an option and instead treat it as part of the filename***, -file07.***



Results: HWasnPhtq9AVKe0dmk45nxy20cvUa6EG

Level 5 → Level 6

Objective

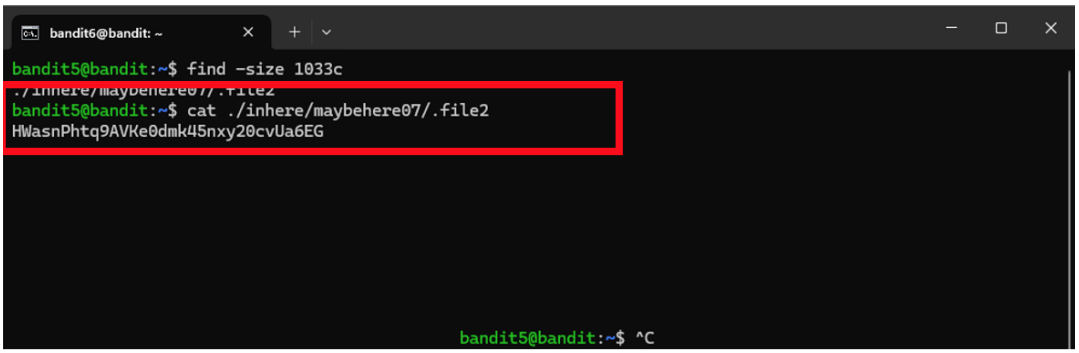
The objective of this level is to find, among the multiple files in the **inhere** directory, one that is human-readable, has a size of 1033 bytes, and is not executable. Inside, we will find the password for the next level, to which we will connect via SSH on port 2220 with the user bandit6.

Methodology

We will use the command ***find -size 1033c*** first to filter out files that do not meet this criterion, and after doing so, it returns only one result.

- **find**: It is a command that allows searching for files and directories in the system based on different criteria.
- **-size 1033c**: Specifies that files with a size of exactly 1033 bytes should be searched for. ***1033*** is the size of the file in bytes, and ***c*** indicates that the unit of measurement is in bytes.

After identifying the only file that meets the size criteria, we will access it using the command cat ***./inhere/maybehere07/.file2***, where we will find the password.



Results: morbNTDkSW6jIlUc0ymOdMaLnOlFVAaj

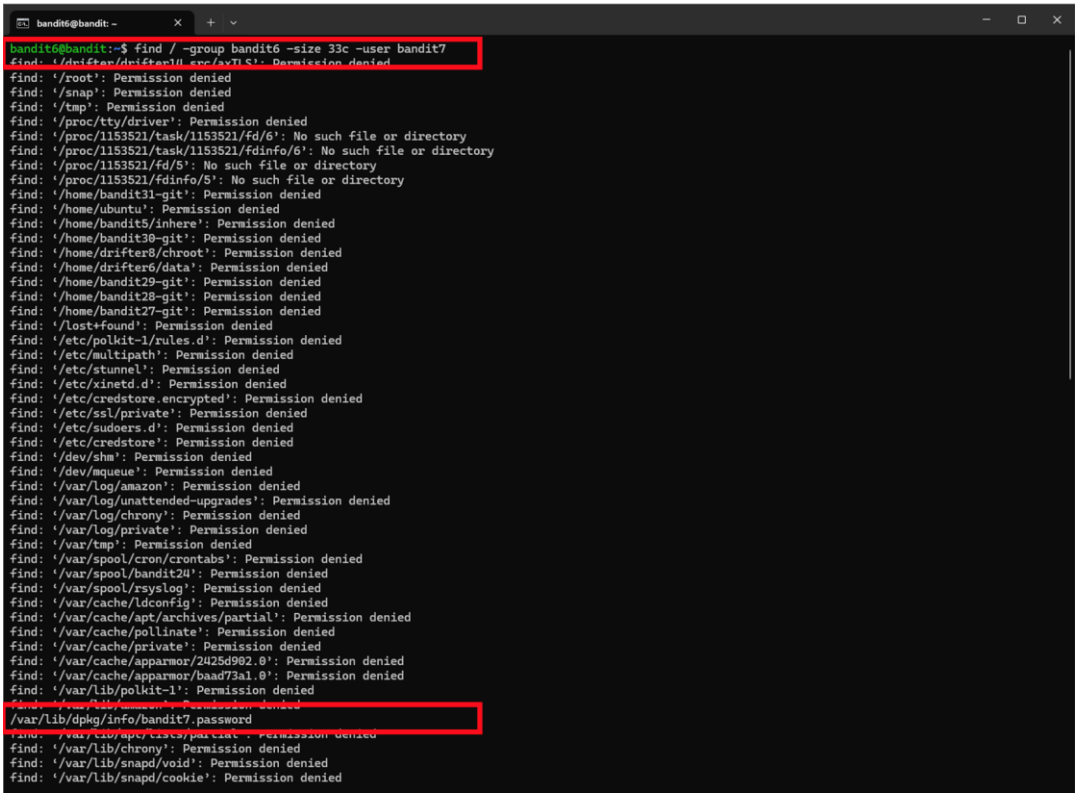
Level 6 → Level 7

Objective

The objective of this level is to find, among the multiple files somewhere on the server, one that is owned by user **bandit7**, owned by group **bandit6**, and has a size of 33 bytes. Inside, we will find the password for the next level, to which we will connect via SSH on port 2220 with user **bandit7**.

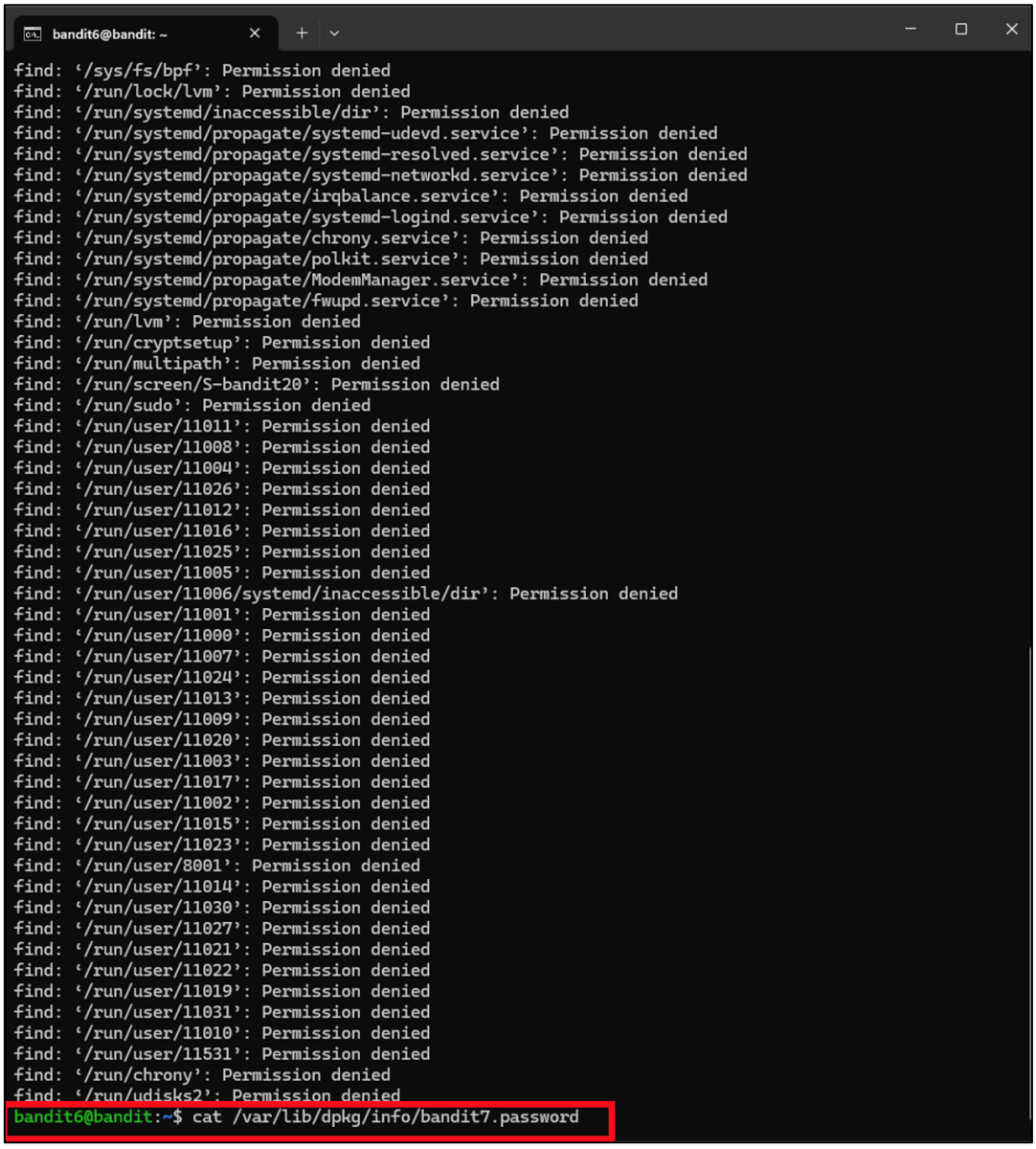
Methodology

We will use the command ***find / -group bandit6 -size 33c -user bandit7*** to filter the files that meet the criteria of the tags. We will then identify the one file that is the only one not showing a "Permission Denied" message, which is ***/var/lib/dpkg/info/bandit7.password***.



- **find /:** Starts the search from the root of the file system (the / directory). It will search throughout the entire file system.
- **-group bandit6:** Filters files that belong to the bandit6 group. It will only display files that have this group as their owner.
- **-size 33c**: Searches for files that are exactly 33 bytes in size. The c indicates that the unit of measurement is in bytes.
- **-user bandit7**: Filters files that belong to the bandit7 user. It will only display files that have this user as their owner.

After identifying the only file that meets the size criteria, we will access it using the command cat ***/var/lib/dpkg/info/bandit7.password***, where we will find the password.



Results: dfwvzFQi4mU0wfNbFOe9RoWskMLg7eEc

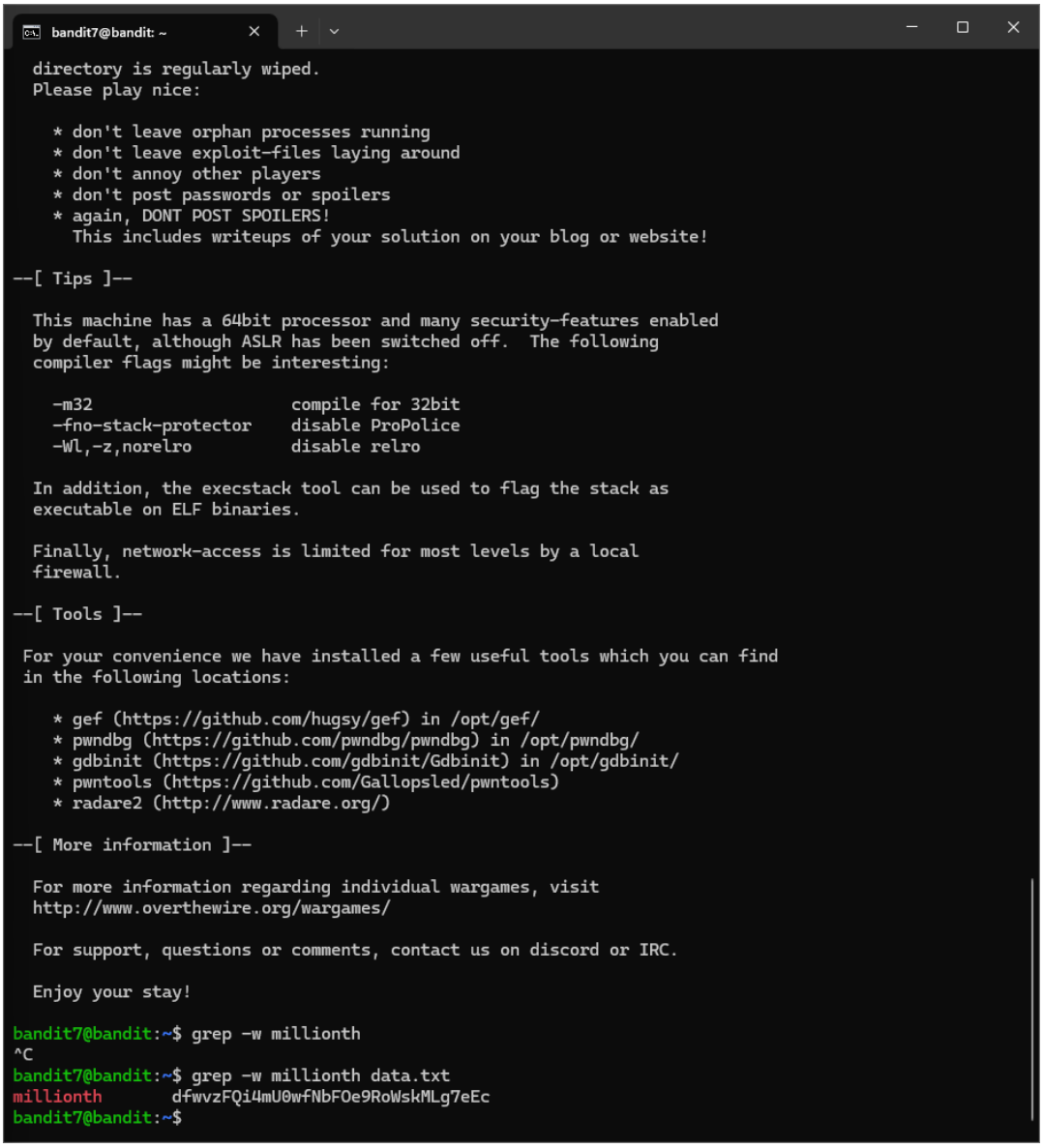
Level 7 → Level 8

Objective

The objective of this level is to find the word millionth that appears before the password, and then connect via SSH on port 2220 with the user bandit8.

Methodology

We will use the command ***grep -w millionth data.txt*** to search for all the lines where the word millionth appears in the data.txt file. When we do this, it will return only one line, and immediately after it, we will find the password.



- **grep**: It is a command used to search for text within a file.
- **-w**: Specifies that **grep** should search for exact matches of the word **millionth**, not partial matches.
- **millionth**: The word we are searching for.

- **data.txt**: The file where the search will be performed.

Results: 4CKMh1JI91bUIZZPXDqGanal4xvAg0JM

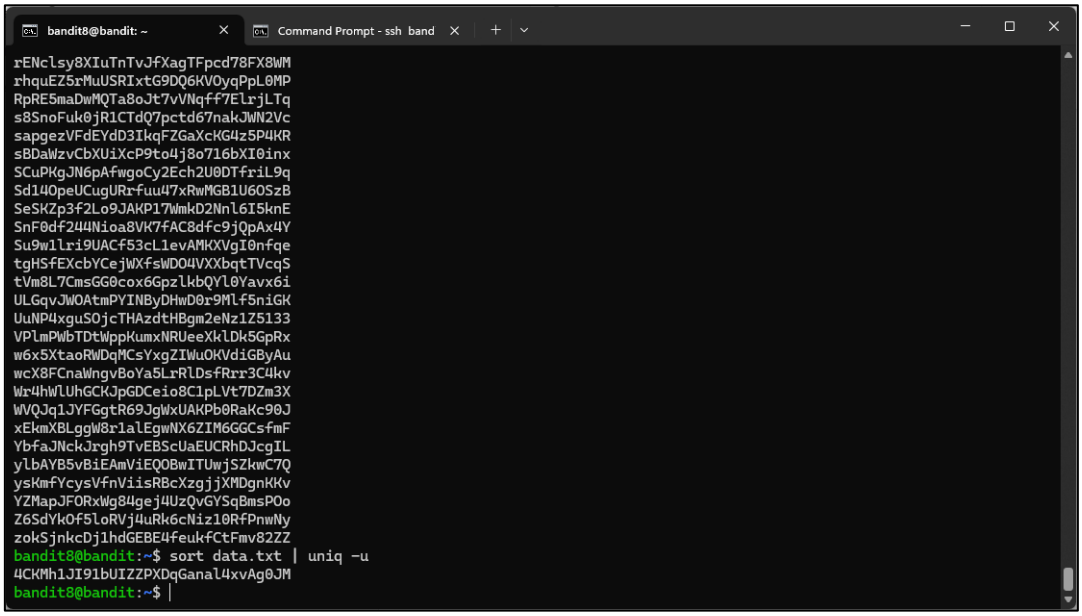
Level 8 → Level 9

Objective

The objective of this level is to find the only line of text that occurs only once, which will be the password that we will use to connect via SSH on port 2220 with the user bandit9.

Methodology

We will be using the command ***sort data.txt | uniq -u*** since we first need to sort the lines and then search for the unique one that does not repeat. After running this command, we will get the password in the console.



- **sort data.txt**: Sorts the data.txt file.
- **uniq -u**: Displays only the lines that occur once in the file.

Results: FGUW5ilLVJrxX9kMYMmlN4MgbpfMiqey

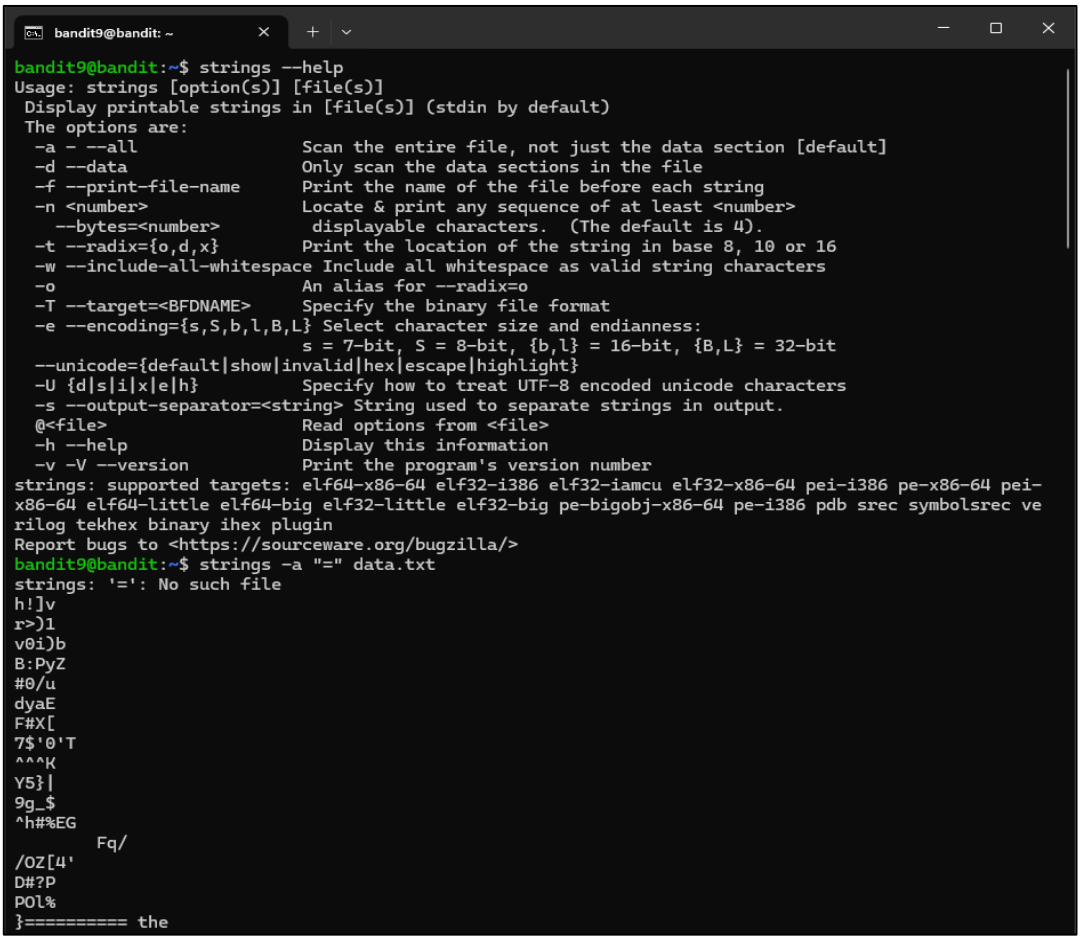
Level 9 → Level 10

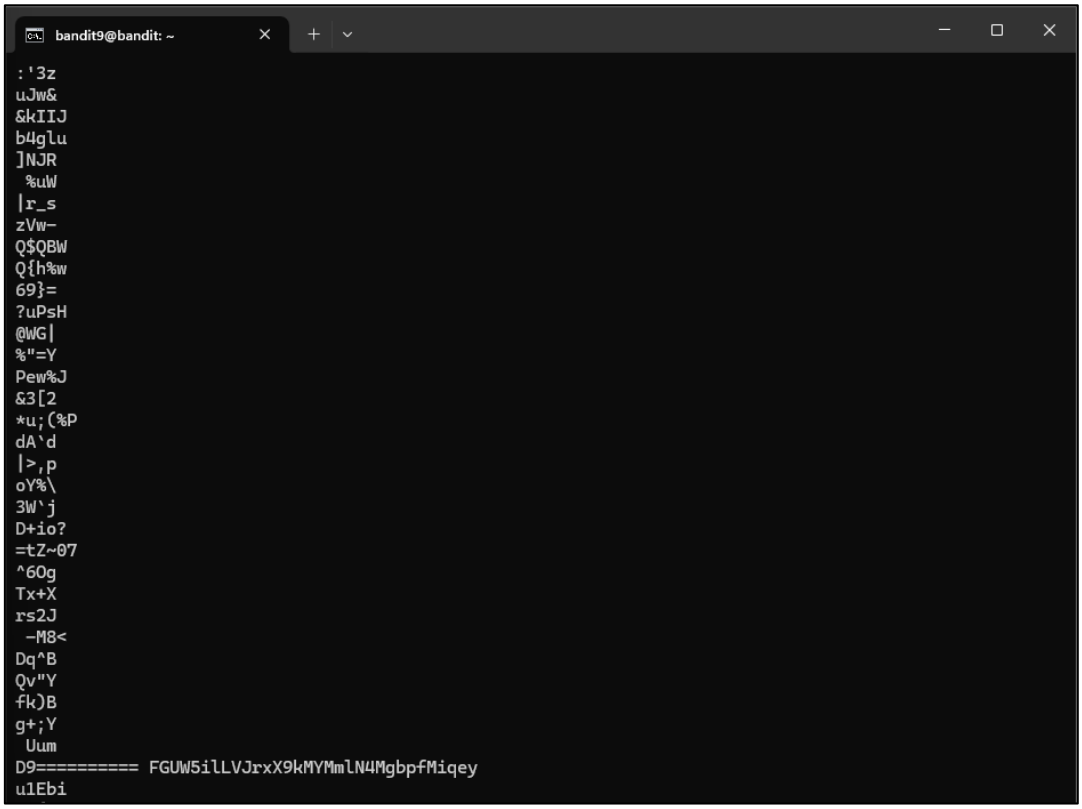
Objective

The objective of this level is to find the password that comes after a long string of '=', which we will use to connect via SSH on port 2220 with the user bandit10.

Methodology

With the command ***strings -a "=" data.txt***, only readable text strings will appear, not binary ones. After this, it was just a matter of searching through the results to find the password.





- **strings data.txt:** This command is used to extract and display printable text strings from the data.txt file.
- **-a:** This option tells the strings command to search through the entire file.
- **"=":** We are searching for the equal sign character "=".

Results: dtR173fZKb0RRsDFSGsg2RWnpNVj3qRr

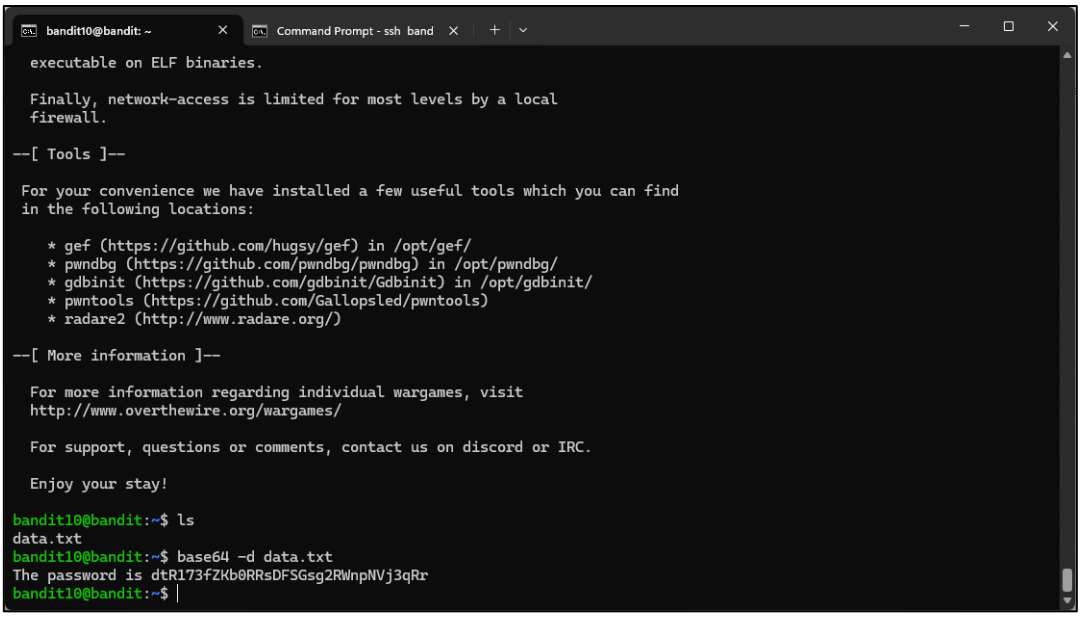
Level 10 → Level 11

Objective

The objective of this level is to find the password for the next level, which is stored in the data.txt file and contains base64 encoded data. We will use this password to connect via SSH on port 2220 with the user bandit11.

Methodology

With the command base64 -d data.txt, decode the content of the file to find the password.



- **base64:** This command is used to encode or decode data in base64 format.
- **-d:** This option tells the base64 command to decode the data.
- **data.txt:** This is the file that contains the base64 encoded data that we want to decode.

Additional Notes

Base64 encoding is a common method used to represent binary data in an ASCII string format. The base64 command in Linux provides a straightforward way to encode and decode such data.

Results: 7x16WNeHIi5YkIhWsfFIqoognUTyj9Q4

Level 11 → Level 12

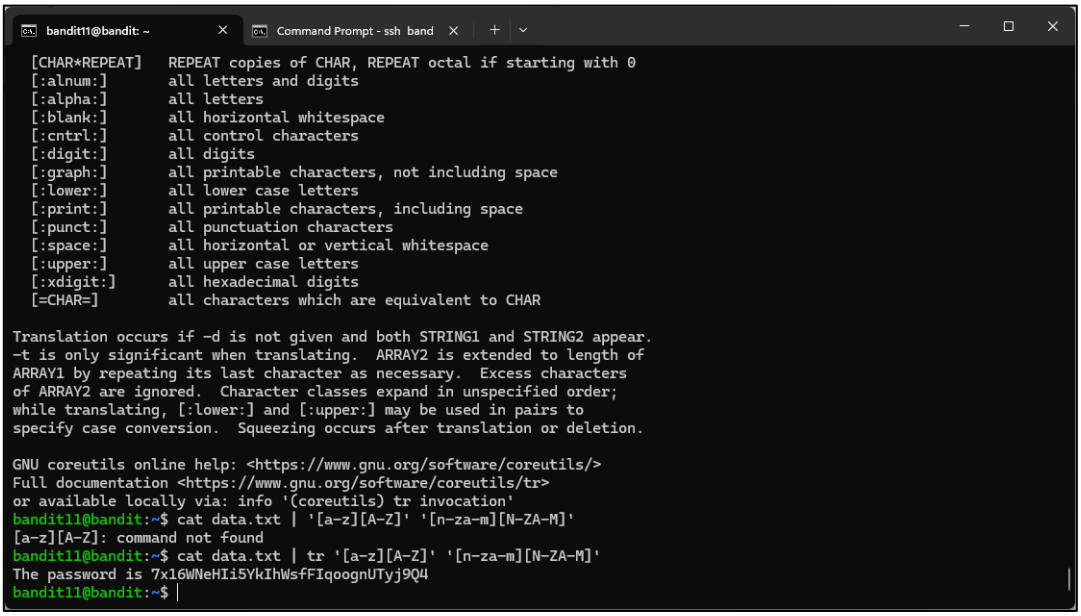
Objective

The objective of this level is to find the password for the next level, which is stored in the file data.txt, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions. We will use this password to connect via SSH on port 2220 with the user bandit12.

Methodology

The ROT13 cipher rotates each letter by 13 positions, so the user employs the tr

command to decode it. The command used is ‘cat data.txt | tr 'A-Za-z' 'N-ZA-Mn-za-m'’



- cat data.txt: Outputs the content of the file.
- tr 'A-Za-z' 'N-ZA-Mn-za-m': The tr (translate) command is used to replace characters based on a specified transformation. In this case, it shifts letters by 13 positions:
    - 'A-Za-z': The input character set includes all uppercase and lowercase letters.
    - 'N-ZA-Mn-za-m': The output character set is the rotated equivalent, with letters shifted by 13 positions. This effectively decodes the ROT13 encoded text, revealing the password.

Results: FO5dwFsc0cbaIiH0h8J2eUks2vdTDwAn

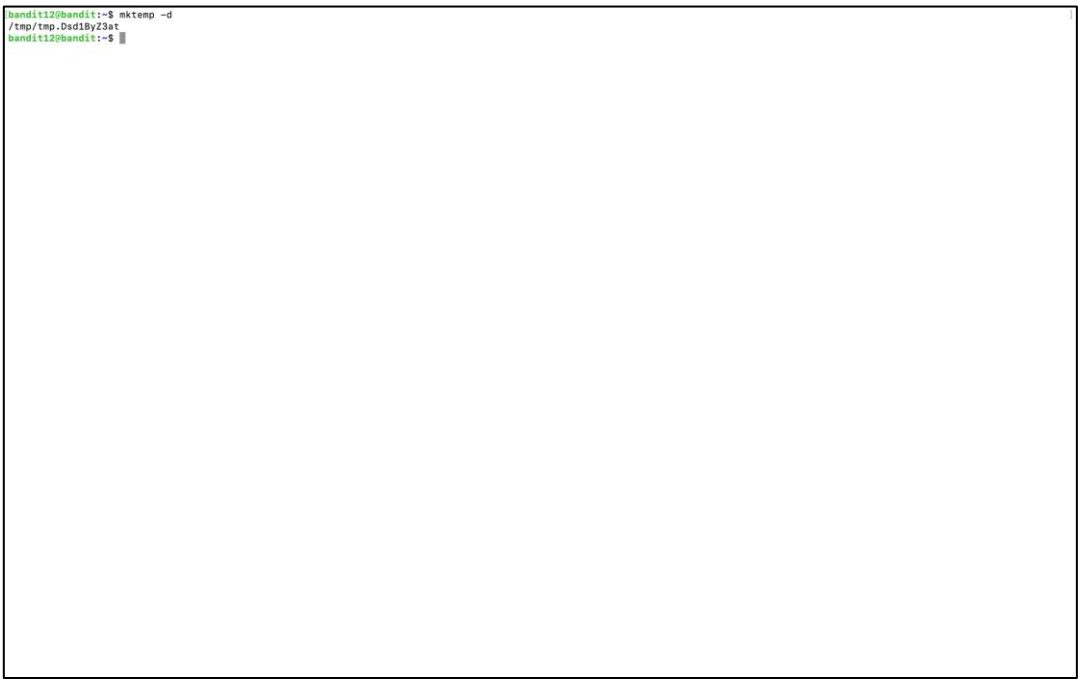
Level 12 → Level 13

Objective

The password for the next level is in **data.txt**, a hexadecimal dump of a repeatedly compressed file. We will need to use different steps to reverse the compression and obtain the original content to connect via SSH to the next level using the password from the file.

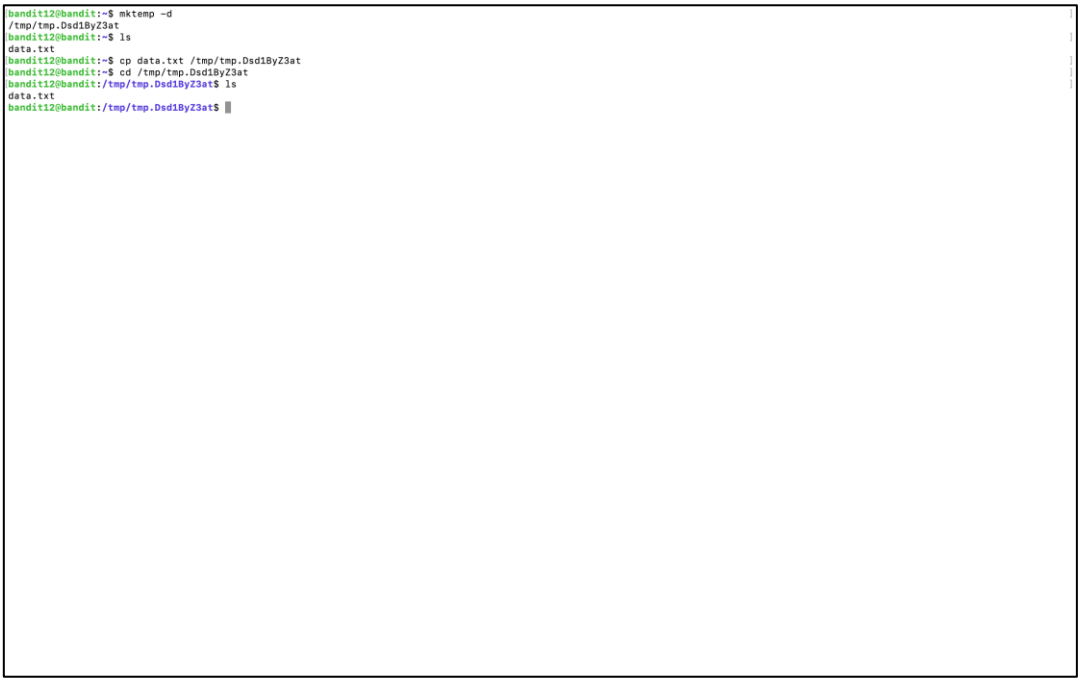
Methodology

The user begins by creating a temporary directory using the mktemp command 'mktemp -d'



mktemp -d: This command creates a new temporary directory with a randomly generated name, which is used to avoid conflicts with other files or directories.

The **data.txt** file located in the root directory will be copied to the temporary folder we created earlier with the following command ***cp data.txt /tmp/tmp.Dsd1ByZ3at***



- **cp:** This command is used to copy files or directories.
- **data.txt:** This is the file being copied.
- **/tmp/tmp.Dsd1ByZ3at:** This is the destination directory, which is a temporary folder created earlier using the mktemp -d command. The randomly generated directory name ensures uniqueness and avoids conflicts with other files or processes.

With the command ***cat data.txt | xxd -r > data***, we will convert the plain text file back to binary format from the hexdump.

```
bandit12@bandit:~$ mktemp -d
/tmp/tmp.Dsd1ByZ3at
bandit12@bandit:~$ ls
data.txt
bandit12@bandit:~$ cp data.txt /tmp/tmp.Dsd1ByZ3at
bandit12@bandit:~$ cd /tmp/tmp.Dsd1ByZ3at
bandit12@bandit:/tmp/tmp.Dsd1ByZ3at$ ls
data.txt
bandit12@bandit:/tmp/tmp.Dsd1ByZ3at$ ls
data.txt
bandit12@bandit:/tmp/tmp.Dsd1ByZ3at$
bandit12@bandit:/tmp/tmp.Dsd1ByZ3at$ cat data | xxd -r > data
bandit12@bandit:/tmp/tmp.Dsd1ByZ3at$ ls
data  data.txt
bandit12@bandit:/tmp/tmp.Dsd1ByZ3at$
```

- xxd -r: Converts the hexdump back into its original binary format.

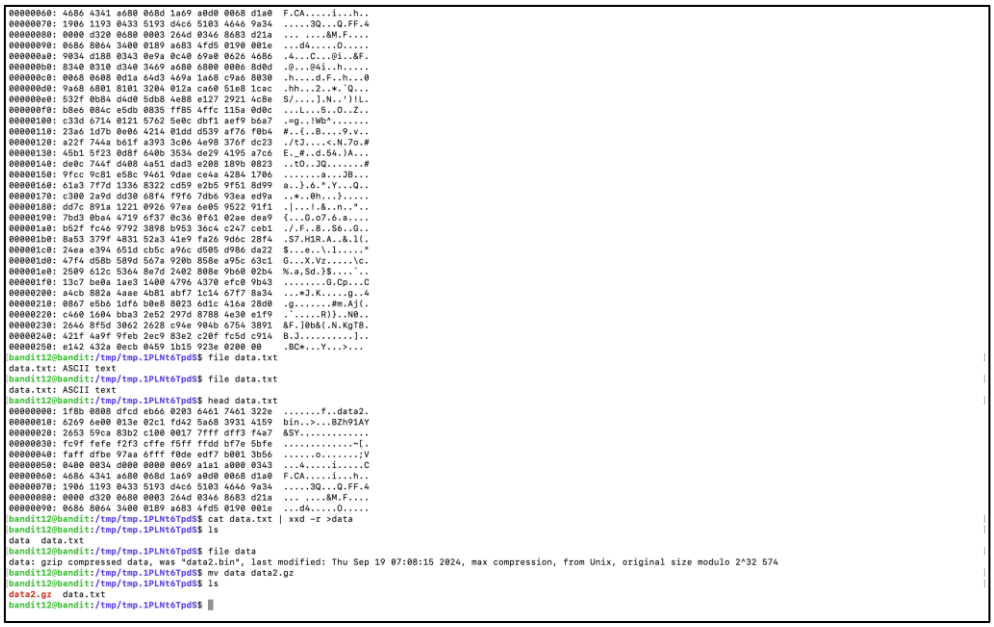
We use the file command at each step to identify the file type: **file data**. We discover that the file is a compressed archive, so we proceed to decompress it using the appropriate commands. This involves renaming files when needed and using the **gzip, bzip2**, and **tar** commands.

❖ Decompressing gzip

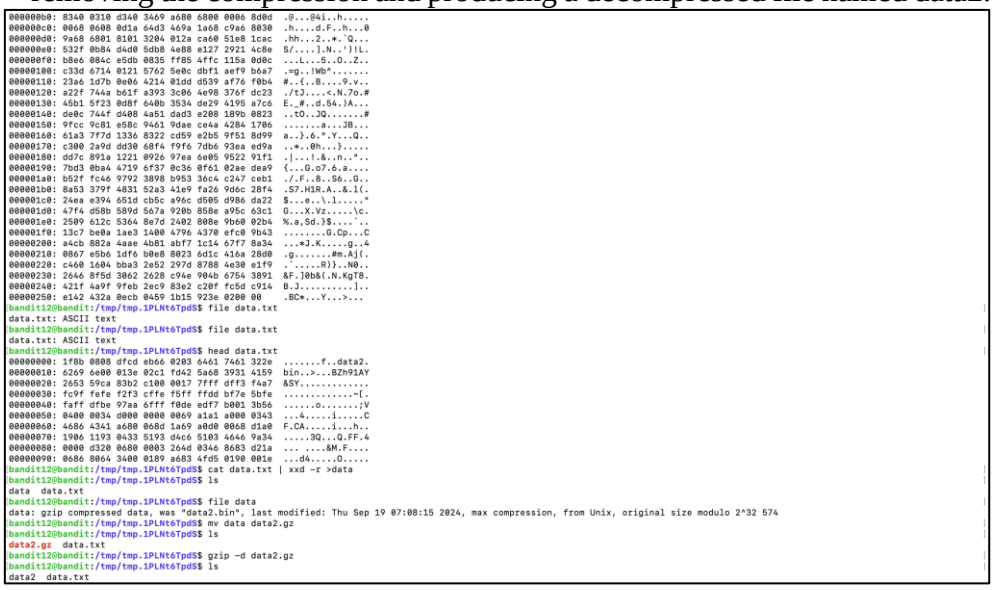
```
000001b0: 8a53 379f 4831 52a3 41e9 fa26 9d6c 28f4  .S7.H1R.A..&.l(.
000001c0: 24ea e394 651d cb5c a96c d505 d986 da22  $...e..\.l.....".
000001d0: 47f4 d58b 589d 567a 920b 858e a95c 63c1  G...X.Vz.....\c.
000001e0: 2509 612c 5364 8e7d 2402 808e 9b60 02b4  %.a,Sd.}$....`..
000001f0: 13c7 be0a 1ae3 1400 4796 4370 efc0 9b43  ........G.Cp...C
00000200: a4cb 882a 4aae 4b81 abf7 1c14 67f7 8a34  ...*J.K.....g..4
00000210: 0867 e5b6 1df6 b0e8 8023 6d1c 416a 28d0  .g.......#m.Aj(.
00000220: c460 1604 bba3 2e52 297d 8788 4e30 e1f9  .`.....R)}..N0..
00000230: 2646 8f5d 3062 2628 c94e 904b 6754 3891  &F.]0b&(.N.KgT8.
00000240: 421f 4a9f 9feb 2ec9 83e2 c20f fc5d c914  B.J..........]..
00000250: e142 432a 0ecb 0459 1b15 923e 0200 00    .BC*...Y...>...
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$ file data.txt
data.txt: ASCII text
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$ file data.txt
data.txt: ASCII text
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$ head data.txt
00000000: 1f8b 0808 dfcd eb66 0203 6461 7461 322e  .......f..data2.
00000010: 6269 6e00 013e 02c1 fd42 5a68 3931 4159  bin..>...BZh91AY
00000020: 2653 59ca 83b2 c100 0017 7fff dff3 f4a7  &SY.............
00000030: fc9f fefe f2f3 cffe f5ff ffdd bf7e 5bfe  ............~[.
00000040: faff dfbe 97aa 6fff f0de edf7 b001 3b56  ......o.......;V
00000050: 0400 0034 d000 0000 0069 a1a1 a000 0343  ...4.....i.....C
00000060: 4686 4341 a680 068d 1a69 a0d0 0068 d1a0  F.CA.....i...h..
00000070: 1906 1193 0433 5193 d4c6 5103 4646 9a34  .....3Q...Q.FF.4
00000080: 0000 d320 0680 0003 264d 0346 8683 d21a  ... ....&M.F....
00000090: 0686 8064 3400 0189 a683 4fd5 0190 001e  ...d4.....O.....
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$ cat data.txt | xxd -r >data
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$ ls
data  data.txt
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$ file data
data: gzip compressed data, was "data2.bin", last modified: Thu Sep 19 07:08:15 2024, max compression, from Unix, original size modulo 2^32 574
bandit12@bandit:/tmp/tmp.1PLNt6TpdS$
```

- **mv data data2.gz:** The mv command renames the file from data to data2.gz. The new extension .gz indicates that the file is compressed using

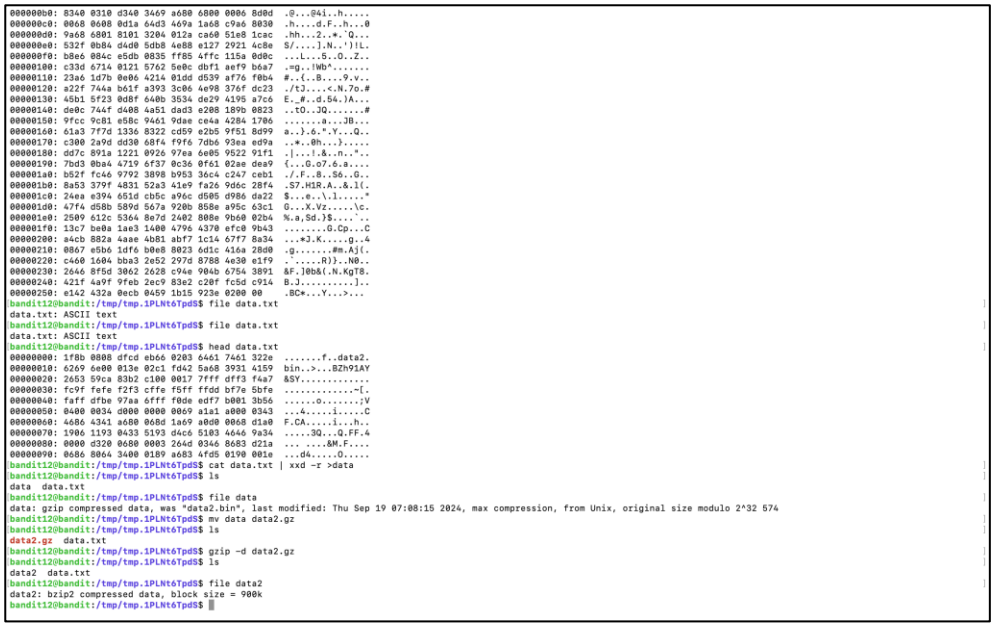
gzip.



- **gzip -d data2.gz:** The gzip -d command decompresses the data2.gz file, removing the compression and producing a decompressed file named data2.



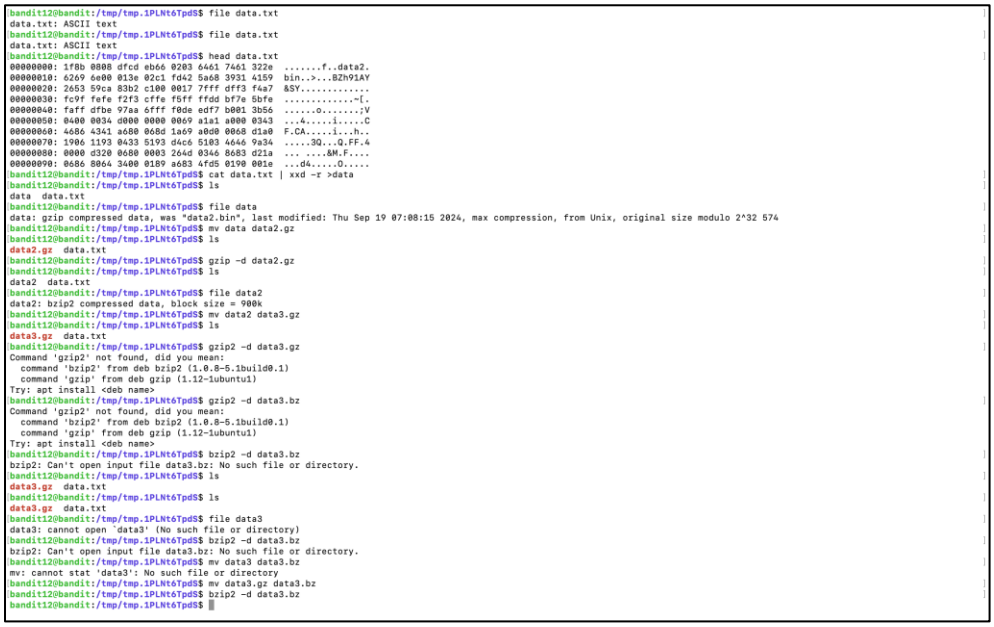
- **file data2:** The file command checks the type of the decompressed file data2 to determine what further processing is needed.



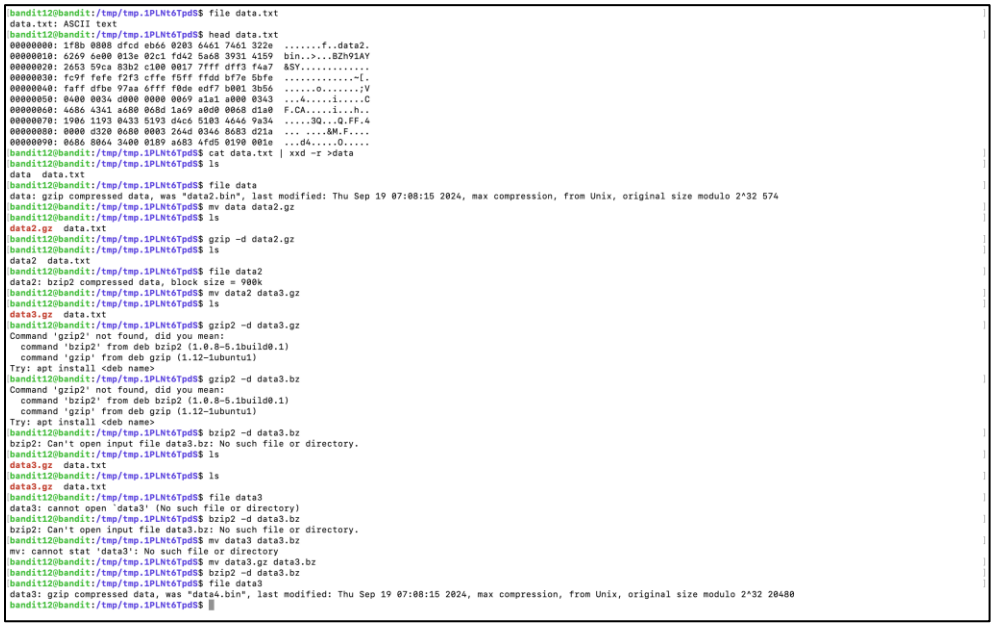
- Decompressing bzip2

  - **mv data2 data3.bz**: The file is renamed from data2 to data3.bz to reflect that it is compressed using bzip2.

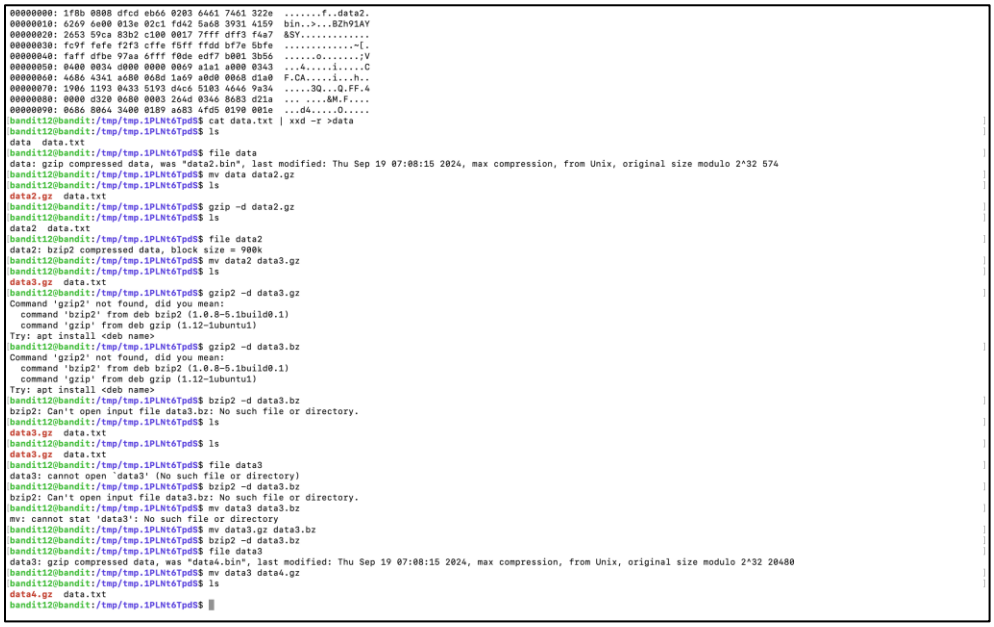
  - **bzip2 -d data3.bz**: The bzip2 -d command decompresses the data3.bz file, producing a decompressed file named data3.



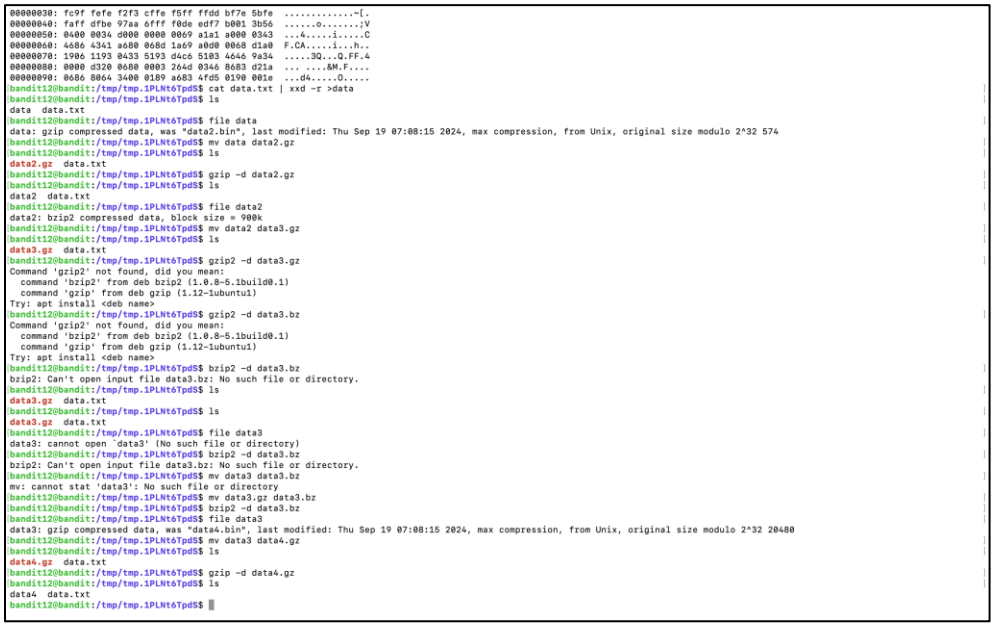
- **file data3**: The file command checks the type of the decompressed file data3 to identify the next processing step.



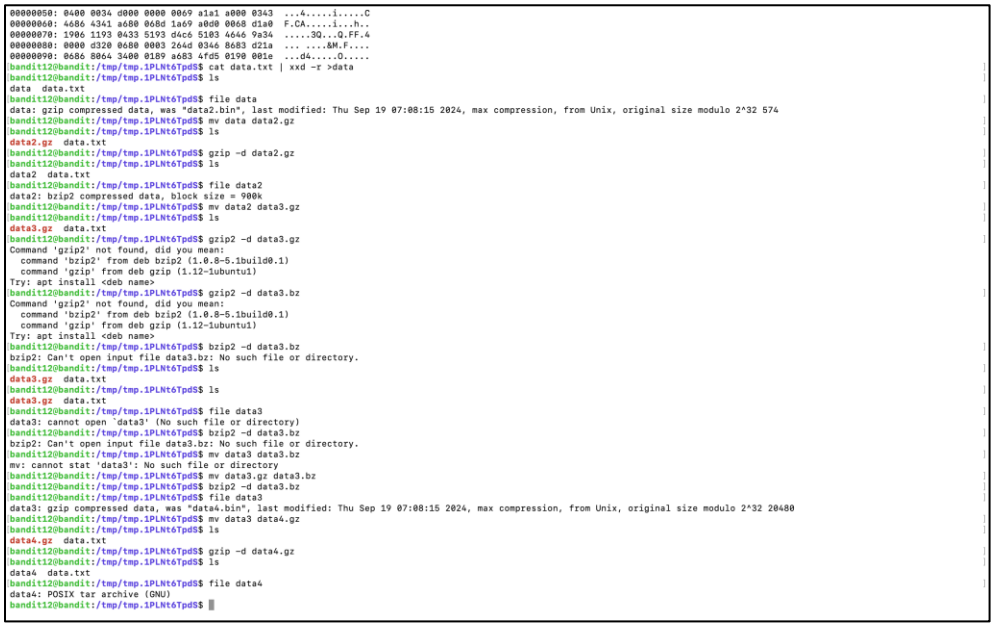
- Decompressing another gzip file
  - **mv data3 data4.gz**: The file is renamed from data3 to data3.gz to reflect that it is compressed using gzip.



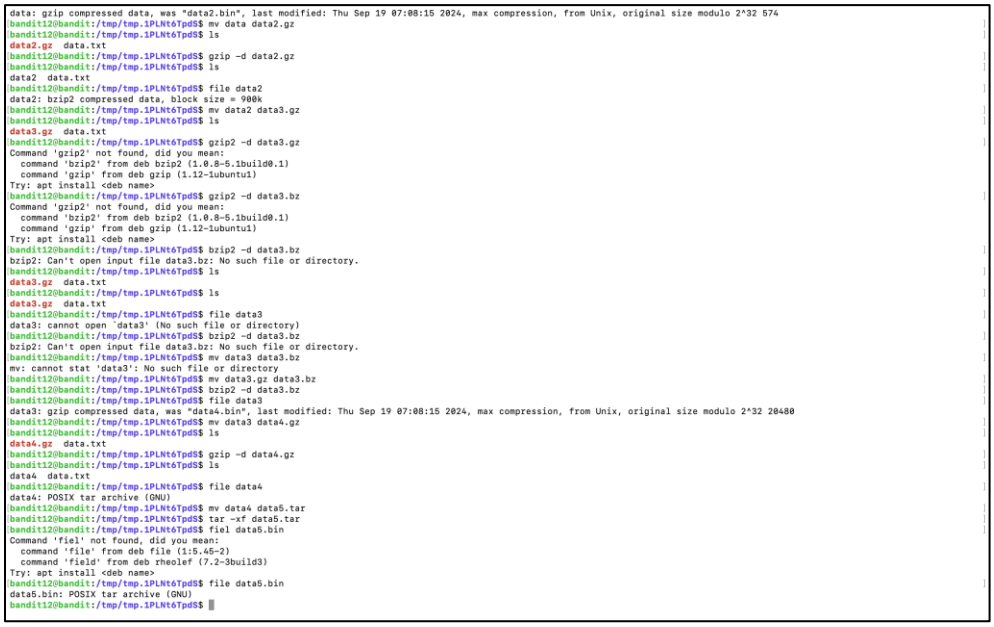
- **gzip -d data4.gz**: The gzip -d command decompresses the data4.gz file, producing a decompressed file named data4.



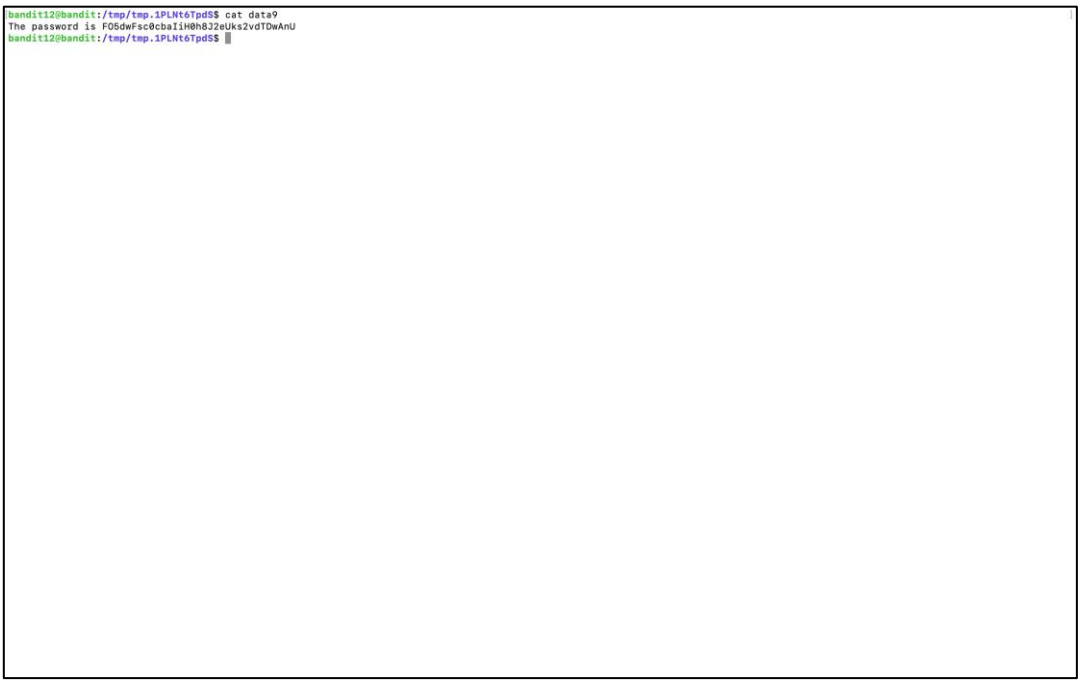
- **file data4**: The file command is used to inspect the type of the file data4.



- Extracting a tar archive
  - **mv data4 data5.tar:** The file is renamed from data4 to data5.tar, reflecting that it is a tar archive.
  - **tar -xf data5.tar**: The tar -xf command extracts the contents of the tar archive into the current directory. The extracted file is named data5.bin.
  - **file data5.bin:** The file command checks the type of the extracted data5.bin file.



We continue decompressing the file following the same steps until we reach the password.



Results: MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS

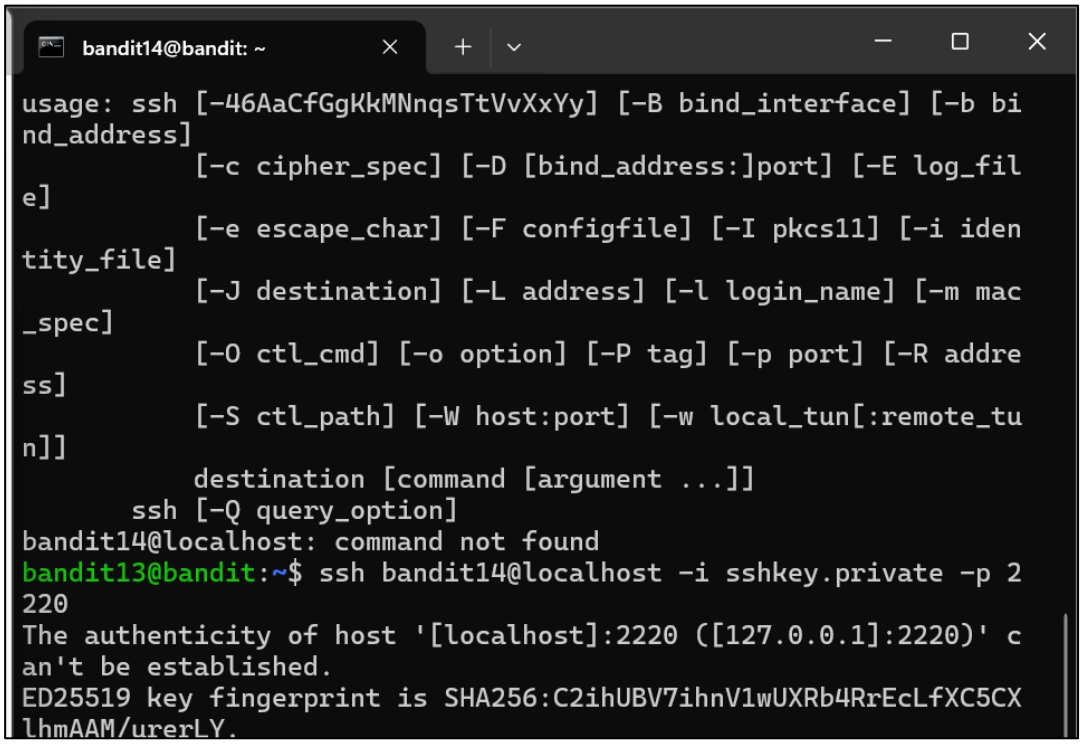
Level 13 → Level 14

Objective

The objective of this level is to find the SSH private key stored in /etc/bandit_pass/bandit14, which can only be read by the user bandit14. With this key, we will be able to connect via SSH on port 2220 using the user bandit14.

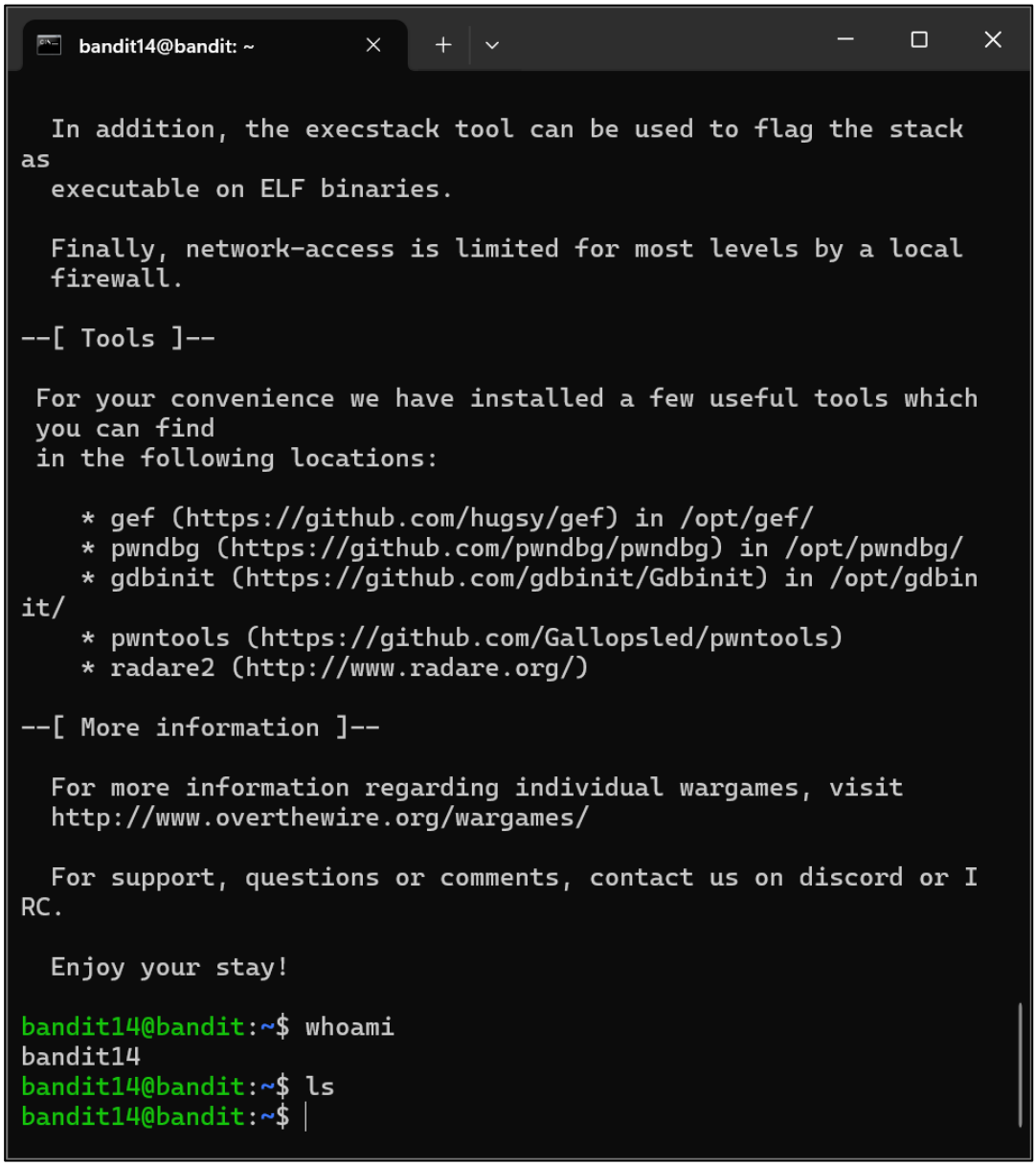
Methodology

We will use the command ***ssh bandit14@localhost -i sshkey.private -p 2220*** to connect to the ***bandit14*** user from the Bandit server.

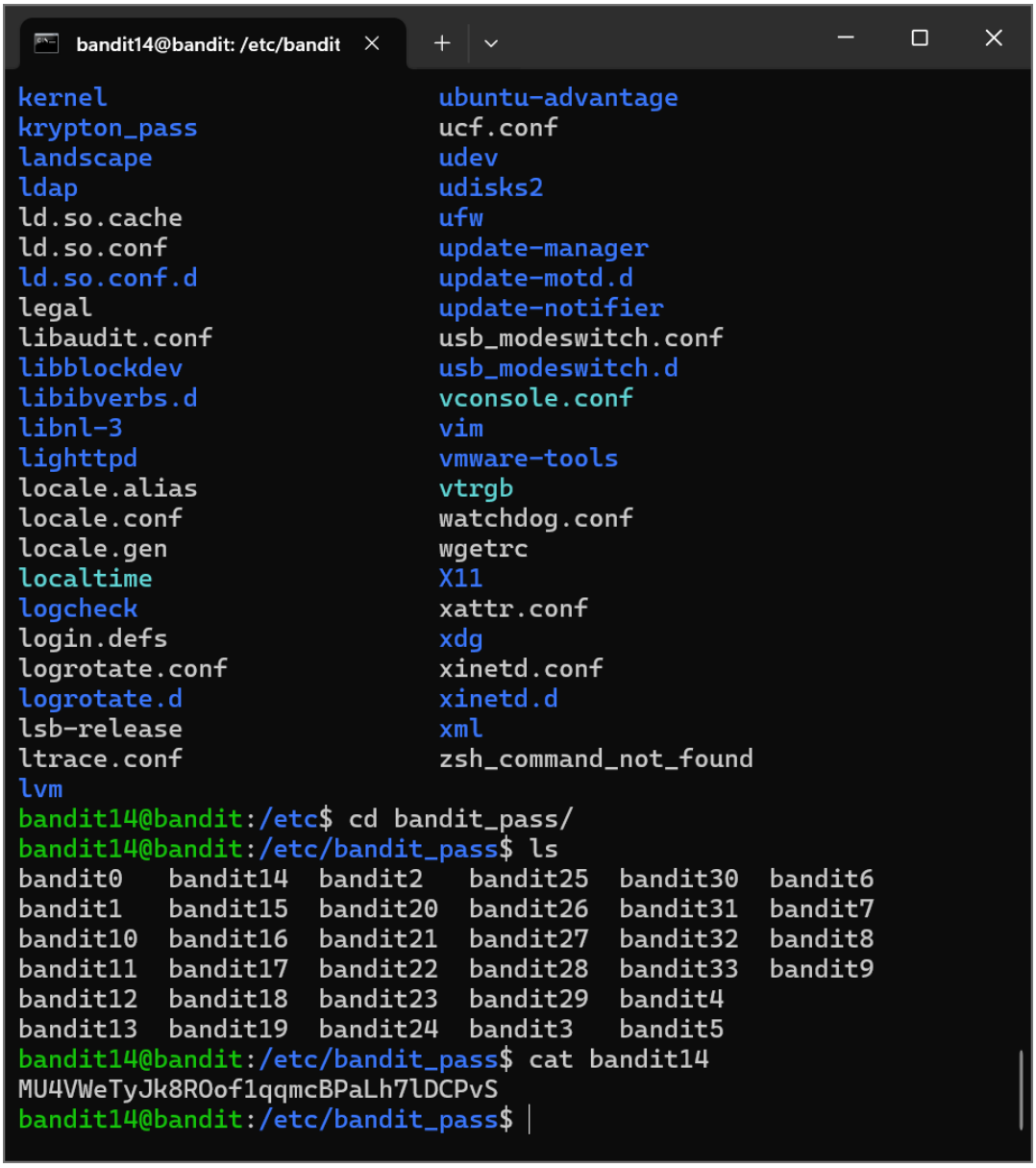


- **ssh bandit14@localhost:** With this command, we start an SSH session with the **bandit14** user, and **localhost** indicates that we are connecting to the same IP address to which we are currently connected.
- **-i sshkey.private:** Specifies the private SSH key to be used for authentication. The key is provided in the sshkey.private file.
- **-p 2220:** Specifies port 2220, which is used throughout the Bandit game for SSH connections.

Use the **whoami** command to verify that you are logged in as bandit14 in order to obtain the password. This should return **bandit14**, confirming that you are logged in with the correct user.



From the root directory, we will locate the folder containing the password files and access the file for bandit14.



Results: 8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo

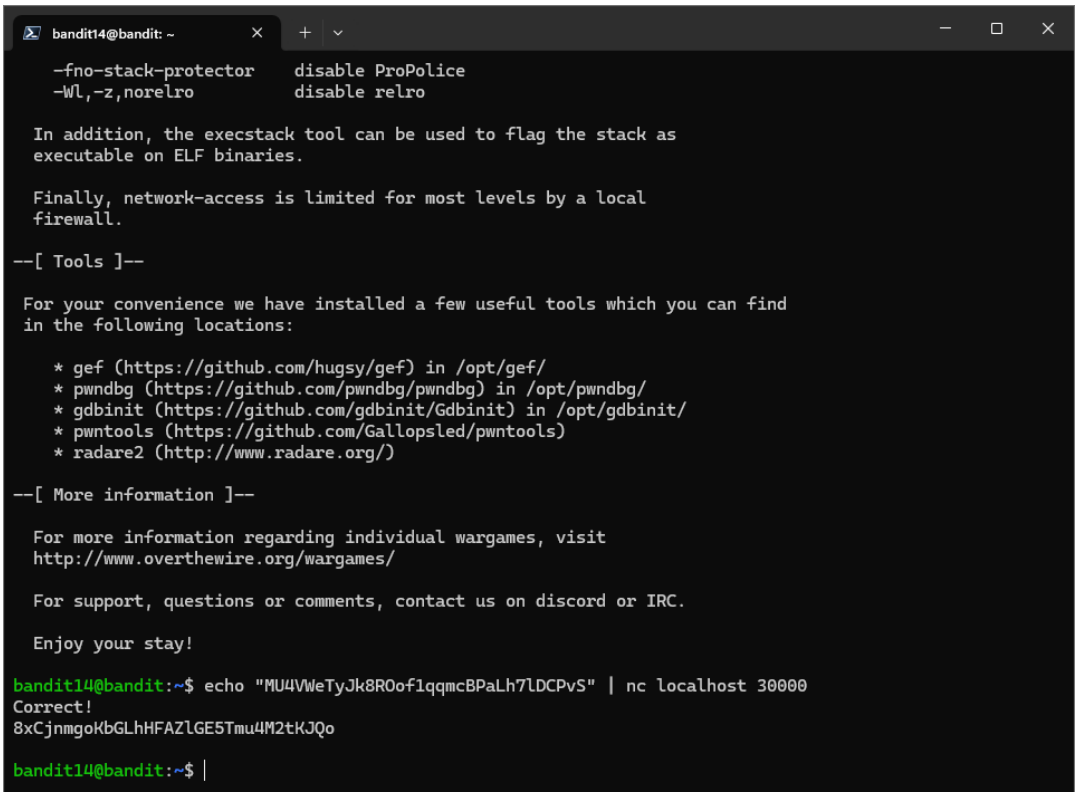
Level 14 → Level 15

Objective

The objective of this level is to send the password from the previous level through port 30000 in order to obtain the password that will allow us to connect via SSH on port 2220 with the user **bandit15**.

Methodology

We will use the command echo "MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS" | nc localhost 30000 to send the previous password to the server.



- ***echo "MU4VWeTyJk8ROof1qqmcBPaLh7lDCPvS":*** Prints the text to the standard output.
- ***|*** **:** Takes the output of the echo command and sends it as input to the next command (nc)
- ***nc localhost 30000:*** (Netcat) is a tool for sending and receiving data over a network. In this case, the connection will be made to the same machine where the command is being executed, and it will connect to the specified port on the server.

Results: 8xCjnmgoKbGLhHFAZlGE5Tmu4M2tKJQo

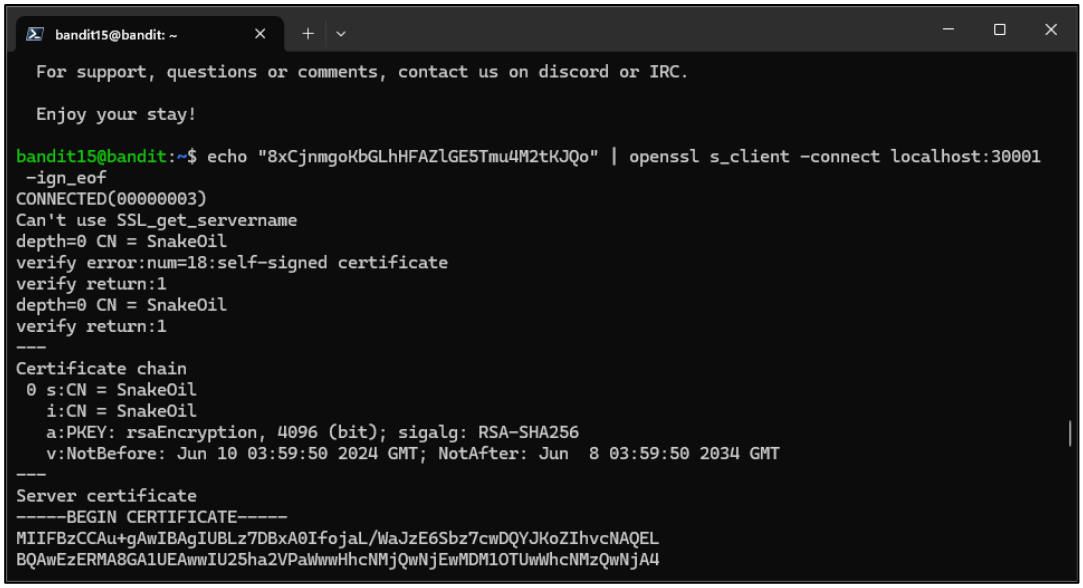
Level 15 → Level 16

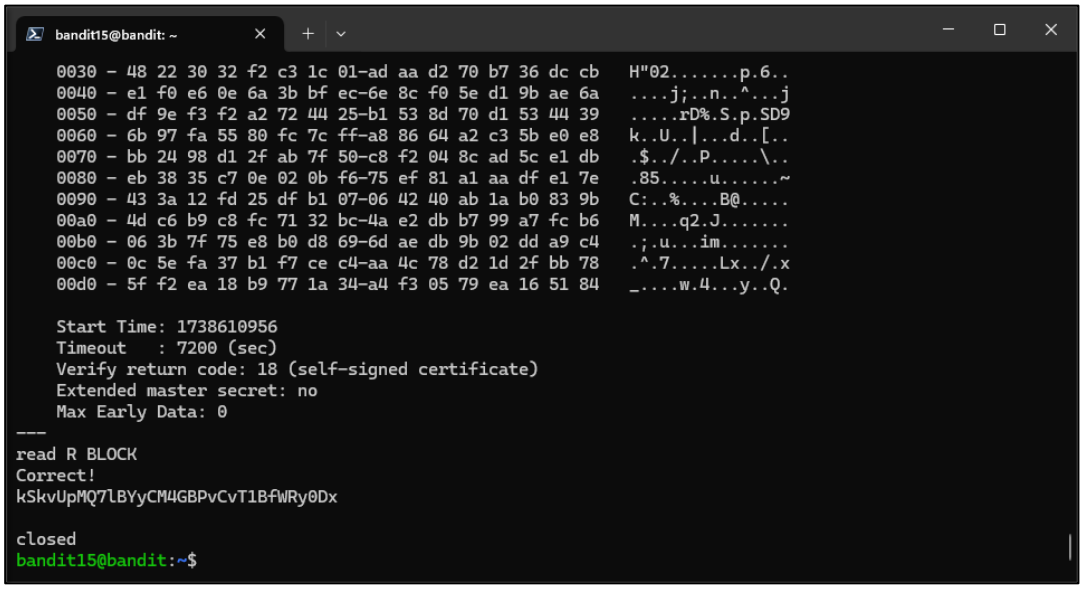
Objective

The objective of this level is to send the password from the previous level encrypted in SSL/TLS through port 30001, in order to obtain the password for the next level and use it to connect via SSH on port 2220 with the user bandit12.

Methodology

For this exercise, we will use part of the previous command to send the password obtained in the previous level to the server, but after the |, the command we will use is openssl s_client -connect localhost:30001.





- openssl: A cryptography and security command.
- s_client: An OpenSSL subcommand that acts as a TLS/SSL client to test secure connections.
- -connect localhost:30001: Specifies the server and port to connect to.

Results: kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx

Level 16 → Level 17

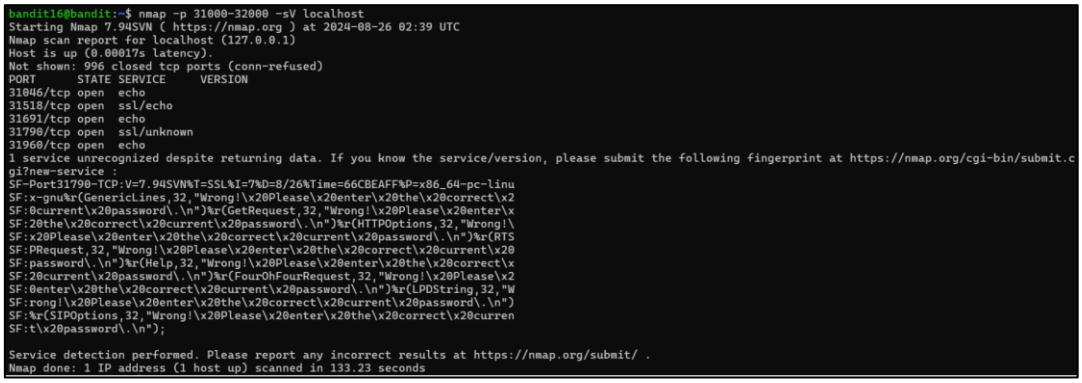
Objective

The objective of this level is to obtain the password for the next level by sending the current level's password to the server through one of the ports ranging from 31000 to 32000 in order to connect via SSH on port 2220 with the user bandit17.

Methodology

To complete level 16 of Bandit, we first scan the port range using nmap to identify which ones are open. We find that five ports are active, including 31518 and 31790, which use SSL. We decide to work with port 31790 and execute a command similar to the one from level 14: echo "password" | openssl s_client -connect localhost:31790 -ign_eof.

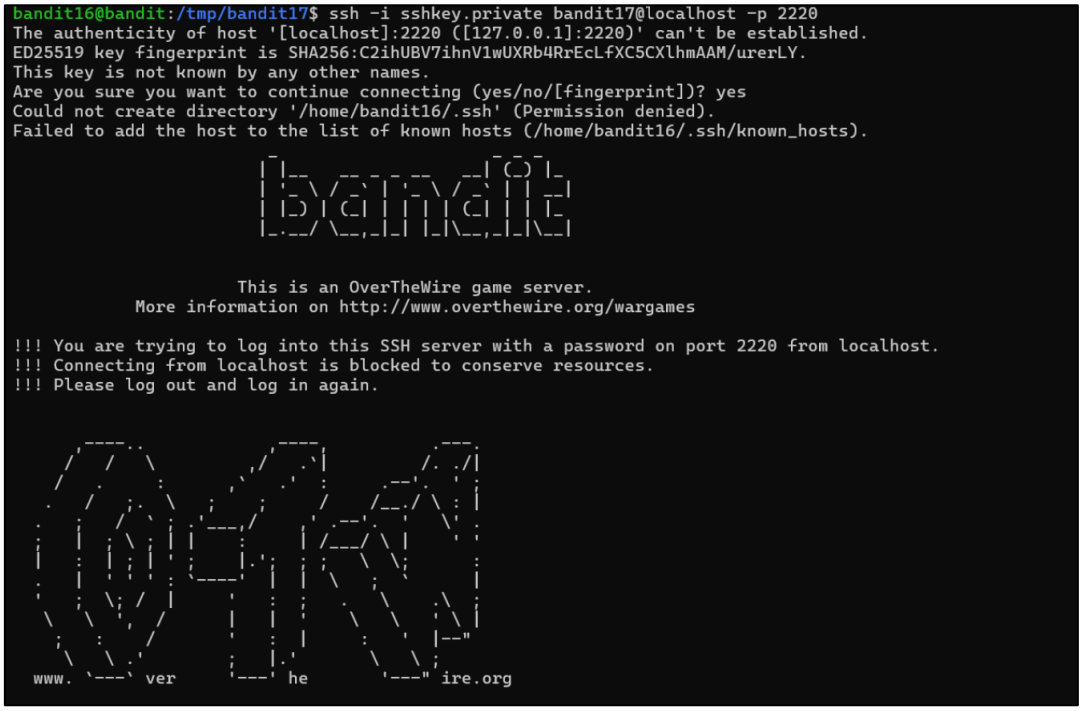
When we run the command, we obtain the server certificate. We copy it and create a directory where we store this information in a file, granting it read and write permissions. Then, we use this information to establish the SSH connection to the next level. Finally, we access the password using the command cat /etc/bandit_pass/bandit17.



```
bandit16@bandit:~$ nmap -p 31000-32000 -sV localhost
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-26 02:39 UTC
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00017s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE     VERSION
31046/tcp open  echo
31518/tcp open  ssl/echo
31691/tcp open  echo
31790/tcp open  ssl/unknown
31960/tcp open  echo
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint at https://nmap.org/cgi-bin/submit.c
gi?new-service :
SF-Port31790-TCP:V=7.94SVN%T=SSL%I=7%D=8/26%Time=66CBEAFF%P=x86_64-pc-linu
SF:x-gnu%r(GenericLines,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x2
SF:0current\x20password\.\n")%r(GetRequest,32,"Wrong!\x20Please\x20enter\x
SF:20the\x20correct\x20current\x20password\.\n")%r(HTTPOptions,32,"Wrong!\
SF:x20Please\x20enter\x20the\x20correct\x20current\x20password\.\n")%r(RTS
SF:PRequest,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x20current\x20
SF:password\.\n")%r(Help,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x
SF:20current\x20password\.\n")%r(FourOhFourRequest,32,"Wrong!\x20Please\x2
SF:0enter\x20the\x20correct\x20current\x20password\.\n")%r(LPDString,32,"W
SF:rong!\x20Please\x20enter\x20the\x20correct\x20current\x20password\.\n")
SF:%r(SIPOptions,32,"Wrong!\x20Please\x20enter\x20the\x20correct\x20curren
SF:t\x20password\.\n");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 133.23 seconds
```

```
bandit16@bandit:~$ echo kSkvUpMQ7lBYyCM4GBPvCvT1BfWRy0Dx | openssl s_client -connect localhost:31790 -ign_eof
CONNECTED(00000003)
Can't use SSL_get_servername
depth=0 CN = SnakeOil
verify error:num=18:self-signed certificate
verify return:1
depth=0 CN = SnakeOil
verify return:1
---
Certificate chain
 0 s:CN = SnakeOil
   i:CN = SnakeOil
   a:PKEY: rsaEncryption, 4096 (bit); sigalg: RSA-SHA256
   v:NotBefore: Jun 10 03:59:50 2024 GMT; NotAfter: Jun  8 03:59:50 2034 GMT
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIFBzCCAu+gAwIBAgIUBLz7DBxA0IfojaL/WaJzE6Sbz7cwDQYJKoZIhvcNAQEL
BQAwEzERMA8GA1UEAwwIU25ha2VPaWwwHhcNMjQwNjEwMDM1OTUwWhcNMzQwNjA4
MDM1OTUwWjATMREwDwYDVQQDDAhTbmFrZU9pbDCCAiIwDQYJKoZIhvcNAQEBBQAD
ggIPADCCAgoCggIBANI+P5QXm9Bj21FIPsQqbqZRb5XmSZZJYaam7EIJ16Fxedf+
jXAv4d/FVqiEM4BuSNsNMeBMx2Gq0lAfN33h+RMTjRoMb8yBsZsC063MLfXCk4p+
09gtGP7BS6Iy5XdmfY/fPHvA3JDEScdlDDmd6Lsbdwhv93Q8M6POVO9sv4HuS4t/
jEjr+NhE+Bjr/wDbyg7GL71BP1WPZpQnRE4OzoSrt5+bZVLvODWUFwinB0fLaGRk
GmI0r5EUOUd7HpYyoIQbiNlePGfPpHRKnmdXTTEZEoxeWWAaM1VhPGqfrB/Pnca+
vAJX7iBOb3kHinmfVOScsG/YAUR94wSELeY+ULEWJaELVUntrJ5HeRDiTChiVQ++
wnnjNbepaW6shopybUF3XXfhIb4NvwLWpvoKFXVtcVjlOujF0snVvpE+MRT0wacy
tHtjZs7Ao7GYxDz6H8AdBLKJW67uQon37a4MI260ADFMS+2vEAbNSFP+f6ii5mrB
18cY64ZaF6oU8bjGK7BArDx56bRc3WFyuBIGWAFHEuB948BcshXY7baf5jjzPmgz
mq1zdRthQB31MOM2ii6vuTkheAvKfFf+llH4M9SnES4NSF2hj9NnHga9V08wfhYc
x0W6qu+S8HUdVF+V23yTvUNgz4Q+UoGs4sHSDEsIBFqNvInnpUmtNgcR2L5PAgMB
AAGjUzBRMB0GA1UdDgQWBBTPo8kfze4P9EgxNuyk7+xDGFtAYzAfBgNVHSMEGDAW
gBTPo8kfze4P9EgxNuyk7+xDGFtAYzAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3
DQEBCwUAA4ICAQAKHomtmcGqyiLnhziLe97Mq2+Sul5QgYVwfx/KYOXxv2T8ZmcR
Ae9XFhZT4jsAOUDK1OXx9aZgDGJHJLNEVTe9zWv1ONFfNxEBxQgP7hhmDBWdtj6d
taqEW/Jp06X+08BtnYK9NZsvDg2YRcvOHConeMjwvEL7tQK0m+GVyQfLYg6jnrhx
egH+abucTKxabFcWSE+Vk0uJYMqcbXvB4WNKz9vj4V5Hn7/DN4xIjFko+nREw6Oa
```



Results: EReVavePLFHtFlFsjn3hyzMlvSuSAcRD

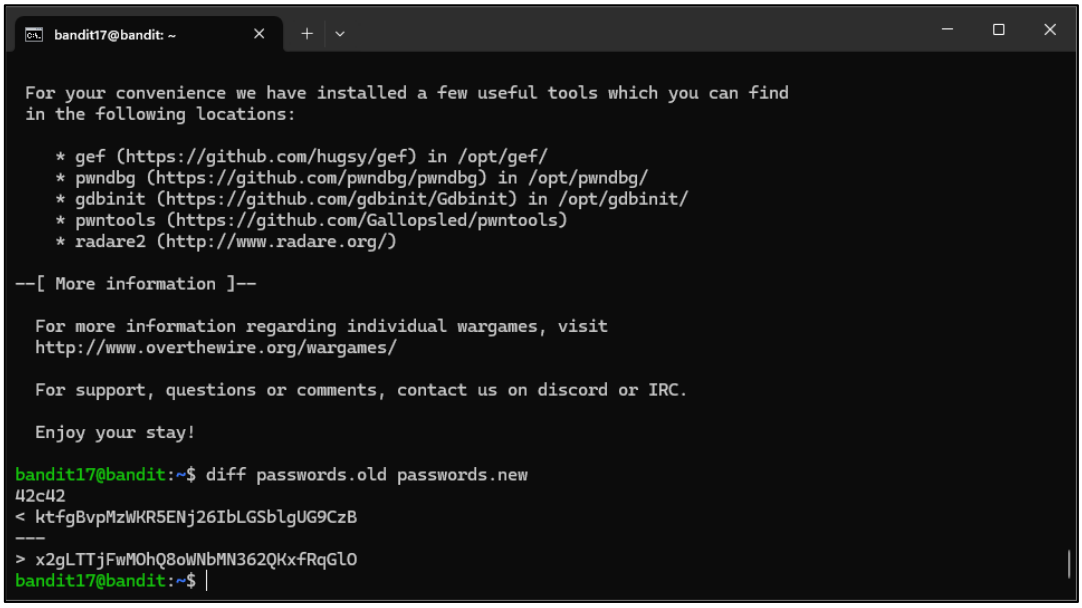
Level 17 → Level 18

Objective

The objective of this level is to find the only line that was changed between the two files (password.old and password.new), which will be the password to connect via SSH on port 2220 with the user bandit18.

Methodology

We will need to compare the two files using the command *diff passwords.new passwords.old*.



- diff: This command shows the differences between two files line by line.
- passwords.new: This is the first file in the comparison.
- passwords.old: This is the second file in the comparison.

Results: x2gLTTjFwMOhQ8oWNbMN362QKxfRqGlO

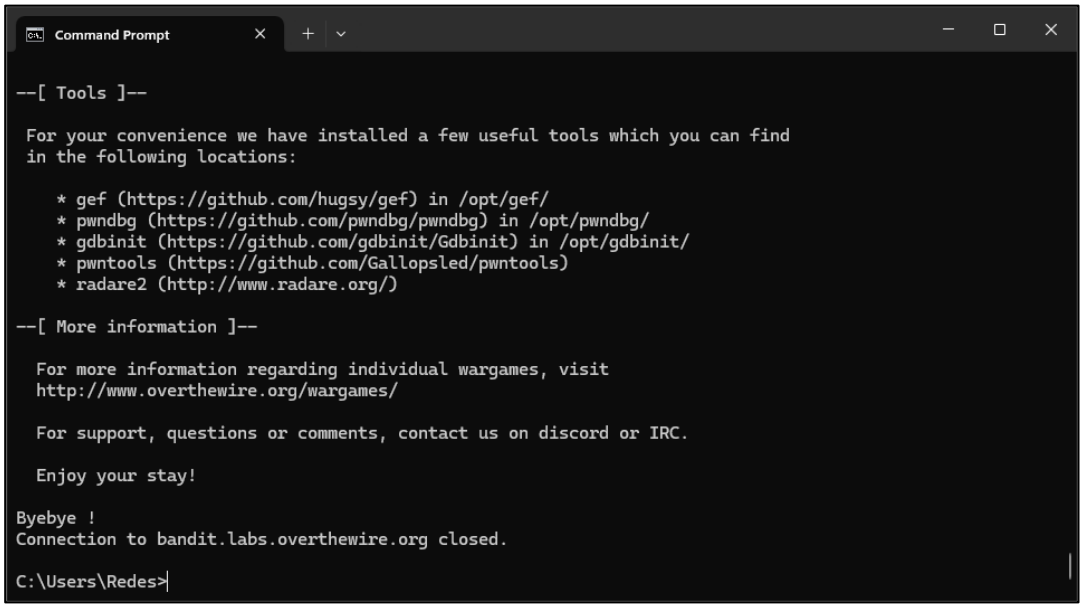
Level 18 → Level 19

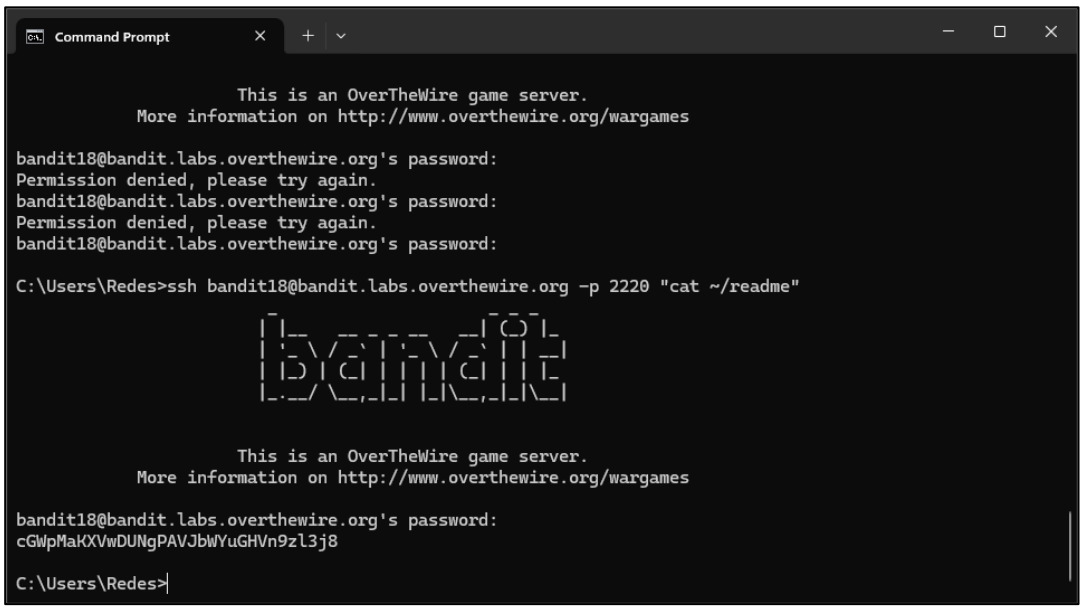
Objective

The objective of this level is to retrieve the password for the next level, which is stored in a file named "readme" located in the home directory. However, when attempting to log in normally via SSH, the session is automatically closed due to a modification in the .bash file.

Methodology

When logging in as we have done in previous levels, after entering the password, a message appears informing us that the connection has been closed. We can solve this by connecting to the server using the command: ***ssh bandit18@bandit.labs.overthewire.org -p 2220 "cat ~/readme"*** , which will give us the password for the next level.





- **ssh:** It is the command to establish a secure connection to a remote server via SSH.
- **bandit18@bandit.labs.overthewire.org:** Specifies the user and the server you are connecting to. In this case, the user is bandit18 and the server is bandit.labs.overthewire.org.
- **-p 2220:** Indicates that the connection should be made through port 2220.
- **"cat ~/readme":** It is the command that will be executed on the server after the connection is established. cat is used to display the content of the readme file located in the user's home directory (represented by ~).

Results: cGWpMaKXVwDUNgPAVJbWYuGHVn9zl3j8

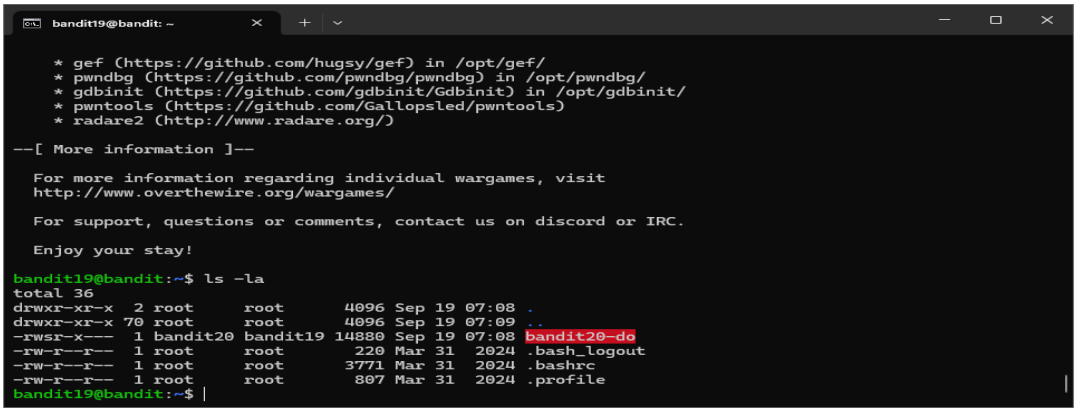
Level 19 → Level 20

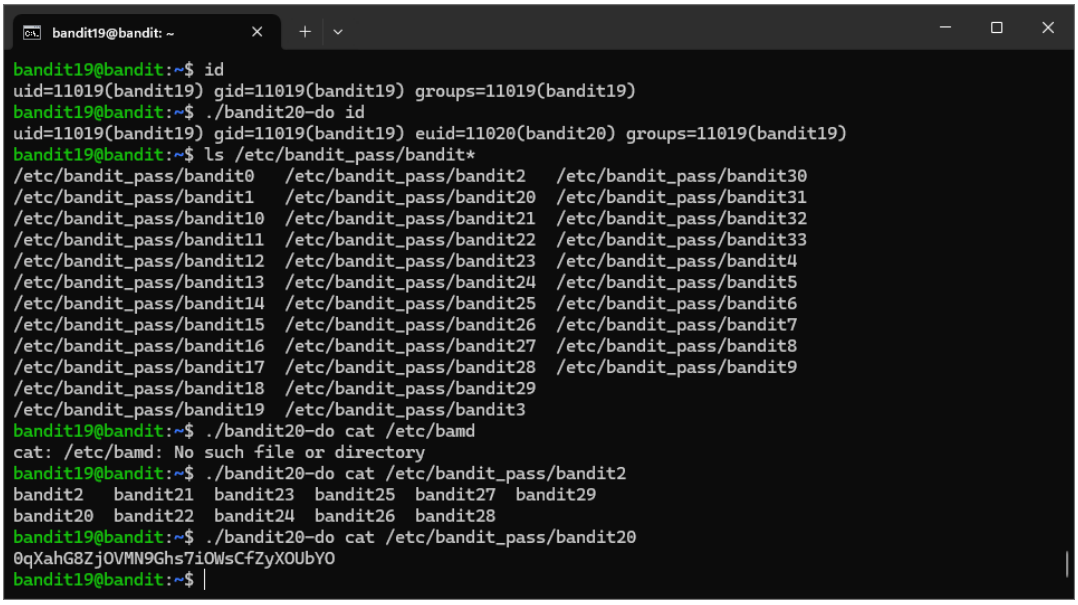
Objective

The objective of this level is to find a way to correctly execute the setuid binary to gain access to the password for the next level. This binary has special configurations that allow certain operations to be executed with elevated privileges, which will enable you to access the password stored in the ***/etc/bandit_pass*** file.

Methodology

First, we need to list the files and directories, showing additional information, including hidden files. Then, with the id command, we display the information about the current user and the groups the user belongs to. Once we have our file **bandit20-do** highlighted previously, we execute this file in the current directory and pass the id as an argument to that file. This way, we will be able to read the password in the specific directory.





- **ls -al:** Displays the file permissions, where the s bit indicates that the binary runs with the permissions of bandit20.
- **./bandit20-do id:** Displays the user and group IDs, showing that the binary runs with the permissions of bandit20.
- **./bandit20-do cat /etc/bandit_pass/bandit20:** Runs cat through the setuid binary to read the restricted password file.

Results: 0qXahG8ZjOVMN9Ghs7iOWsCfZyXOUbYO

Level 20 → Level 21

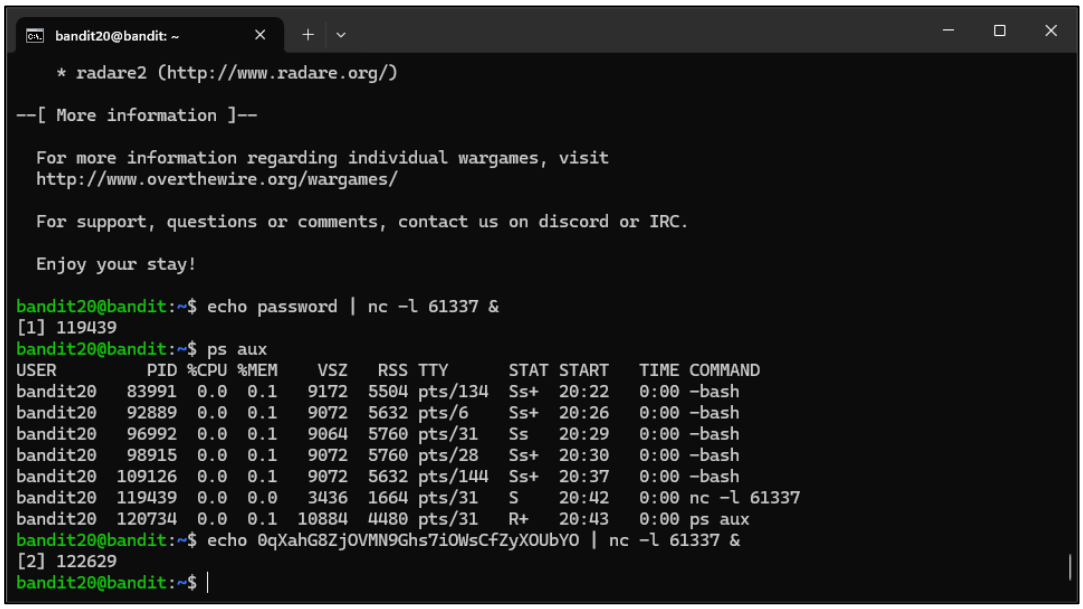
Objective

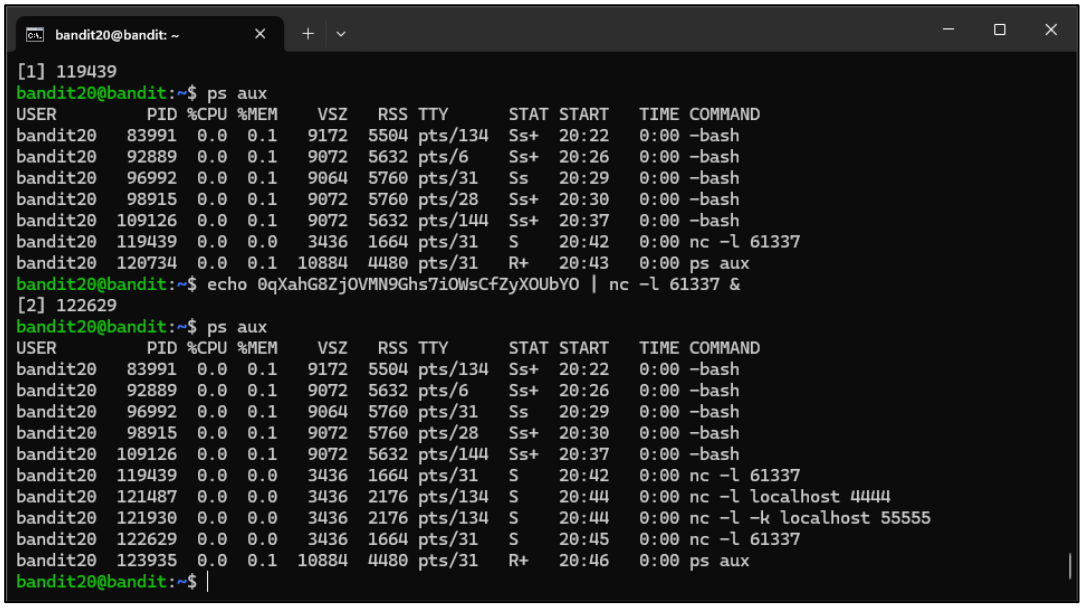
The objective of this level is to use a setuid binary in the home directory that connects to localhost on a port specified as a command-line argument. The binary reads a line of text from the connection and compares it with the password from the previous level (bandit20). If the password is correct, the binary will transmit the password for the next level (bandit21).

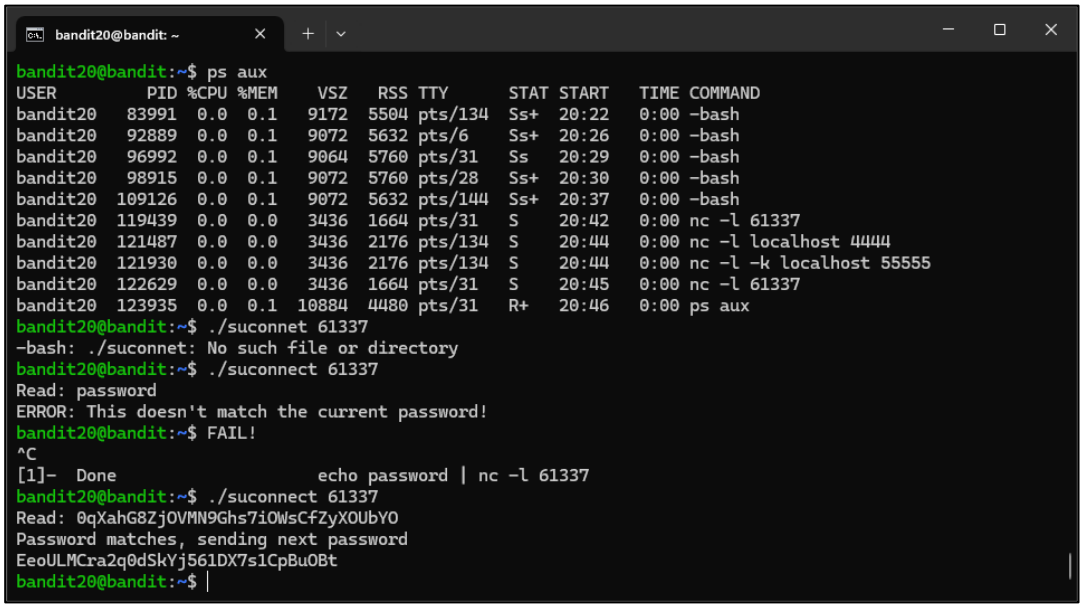
Methodology

We will use the command **echo password | nc -l 61337 &** to start the local network server and listen on a specific port, simulating the network service to which the binary will connect. Next, we will ensure that the setuid binary can establish the connection by using the command to view all running processes, confirming that the

service is active on port 61337. The setuid binary is used to connect to the local network service and retrieve the password for the next level. By executing **./suconnect 61337**, the binary connects to port 61337, sends the current password, and if correct, displays the password for the next level.







- **nc -l 61337**: Passes the output to the nc (netcat) command, which starts a network listener on port 61337.
- **&**: Runs the network listener in the background, allowing the terminal to be used for other commands.
- **ps aux**: Displays all running processes.
- **./suconnect 61337**: Executes the setuid binary suconnect with the port number 61337 as an argument.

Results: EeoULMCra2q0dSkYj561DX7s1CpBuOBt

Level 21 → Level 22

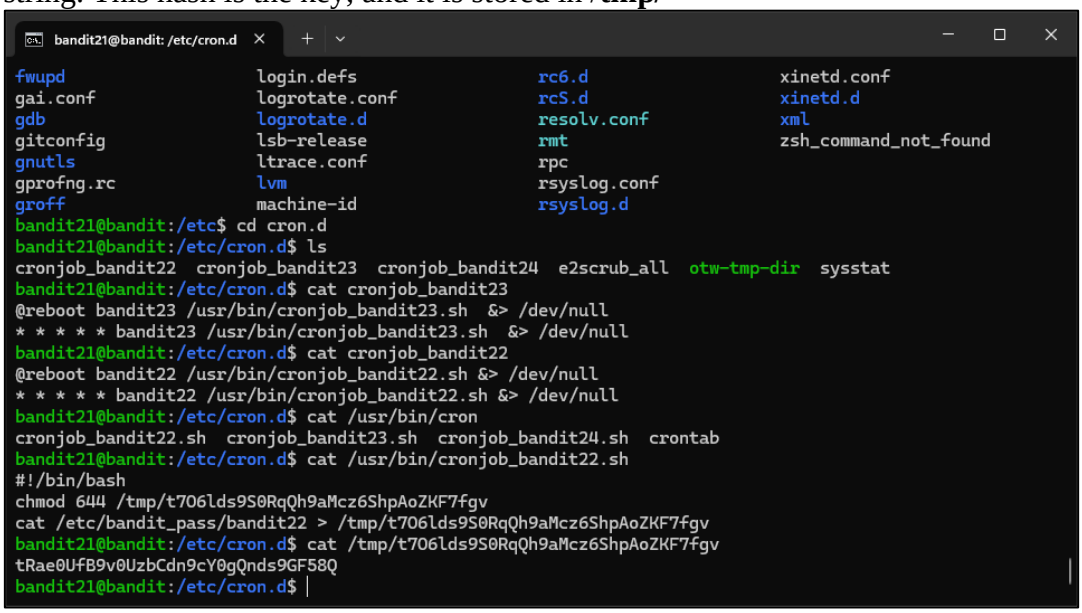
Objective

The objective of this level is to identify the command that is being executed automatically at regular intervals by cron, the time-based job scheduler. To achieve this, it is necessary to examine the configuration in the **/etc/cron.d/** directory and find the specific command that is being executed.

Methodology

You need to identify a cron job that runs automatically on the system. Start by listing and accessing the **/etc/cron.d/** directory, where cron jobs are configured. Then, inspect the **cronjob_bandit** file, which defines a specific cron job that

executes a script located at **/usr/bin/cronjob_bandit23.sh**. By reviewing the contents of this script, you can see that it generates an MD5 hash based on a specific string. This hash is the key, and it is stored in /**tmp/**



Results: tRae0UfB9v0UzbCdn9cY0gQnds9GF58Q

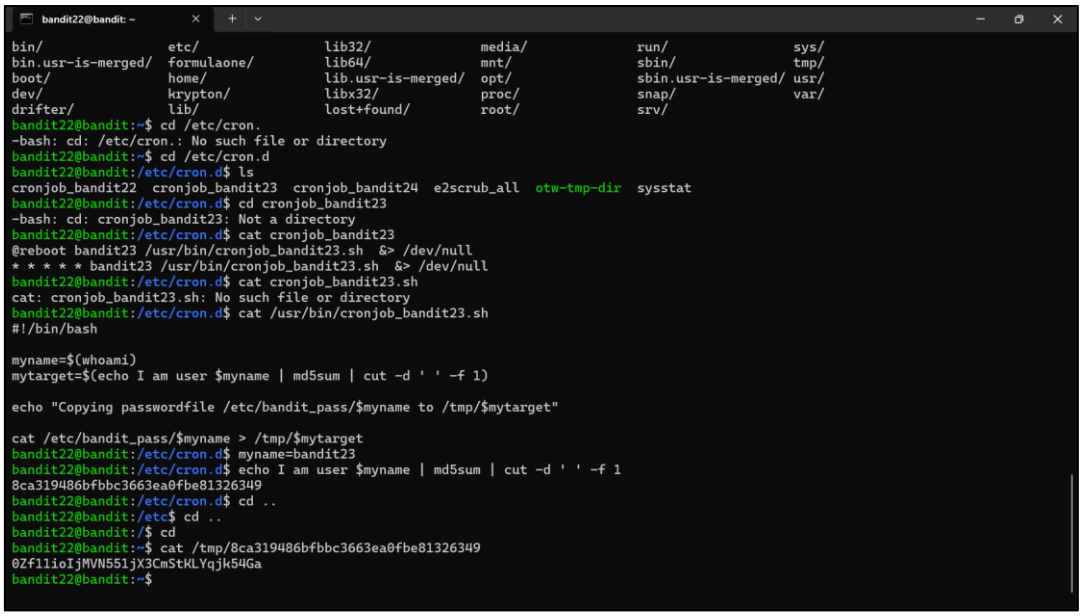
Level 22 → Level 23

Objective

The objective of this level is to find the password for the next level, this by analyzing the shell that is running at regular intervals.This means that we need to examine the cron job configuration and associated script.

Methodology

To find the password we are going to go the directory that contain the shell and by using the comand *cat cronjob_bandit23.sh*



As we can see in the image above, we first saw how the shell works. With the comand that the shell give to us we put it with an asignation of *myname=bandit23 (that is the current user).* Then we shaw that the code that display was a file, so we do a cat command and the password was there.

mytarget=$(echo I am user $myname | md5sum | cut -d ' ' -f 1): Creates an MD5 hash of the string "I am user $myname" and extracts the hash using cut. This hash is assigned to the variable mytarget.

Result**: 0Zf11ioIjMVN551jX3CmStKLYqjk54Ga**

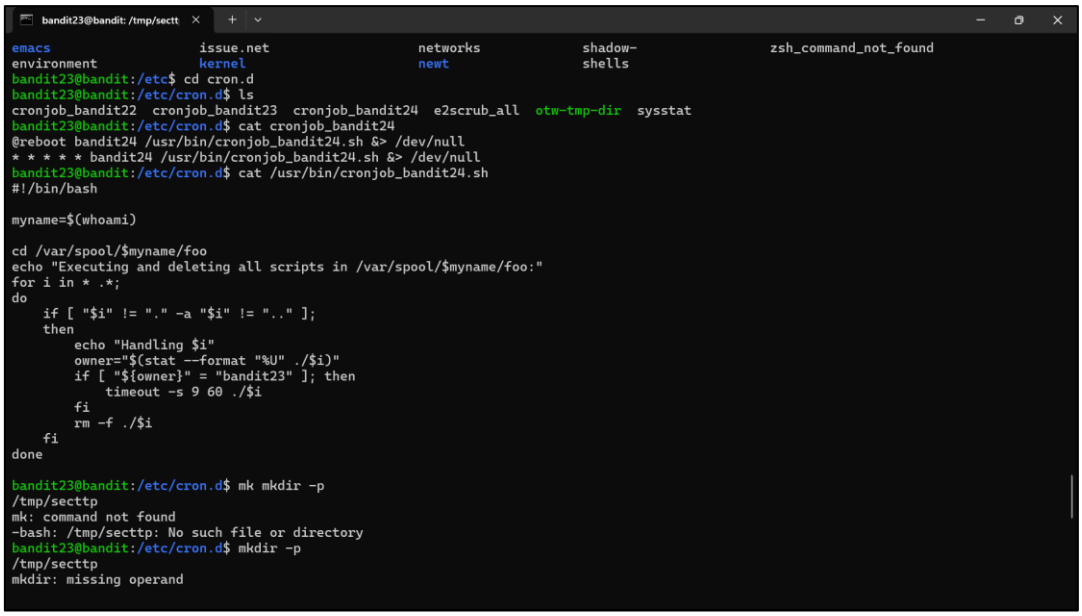
Level 23 → Level 24

Objective

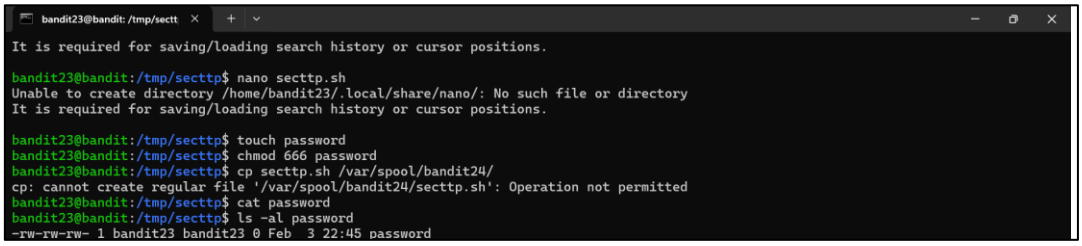
The objective of this level is to find the password for the next level, this by analyzing the shell that is running at regular intervals.This means that we need to examine the cron job configuration and associated script. Also, we need to understand how cron jobs execute scripts and how them mange things.

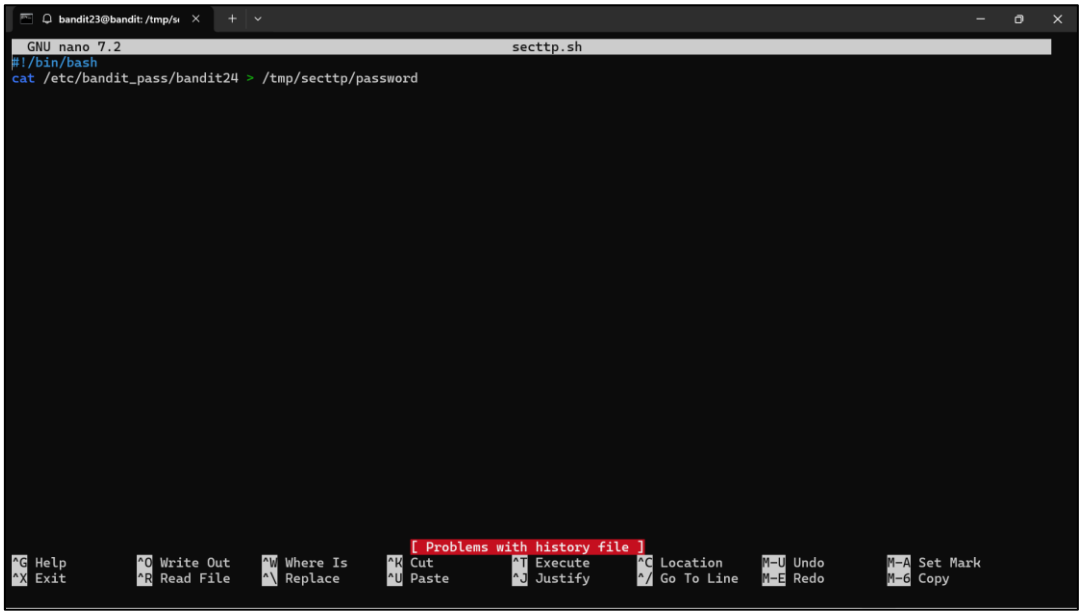
Methodology

To find the password we are going to go the directory that contain the shell and by using the comand *cat /etc/cron.d/cronjob_bandit24.sh*. In order to saw the content of the bandit 23 cron job.



Then we have to make some configuration for copying the file that has the password in which we don't have access to into other in which we do have access to. So we follow the following steps: To create a shell script that will be executed by the cron job, follow these steps 'mkdir -p /tmp/secttp', 'cd /tmp/secttp', 'touch secttp.sh', 'chmod 777 secttp.sh' and 'vim secttp.sh'.

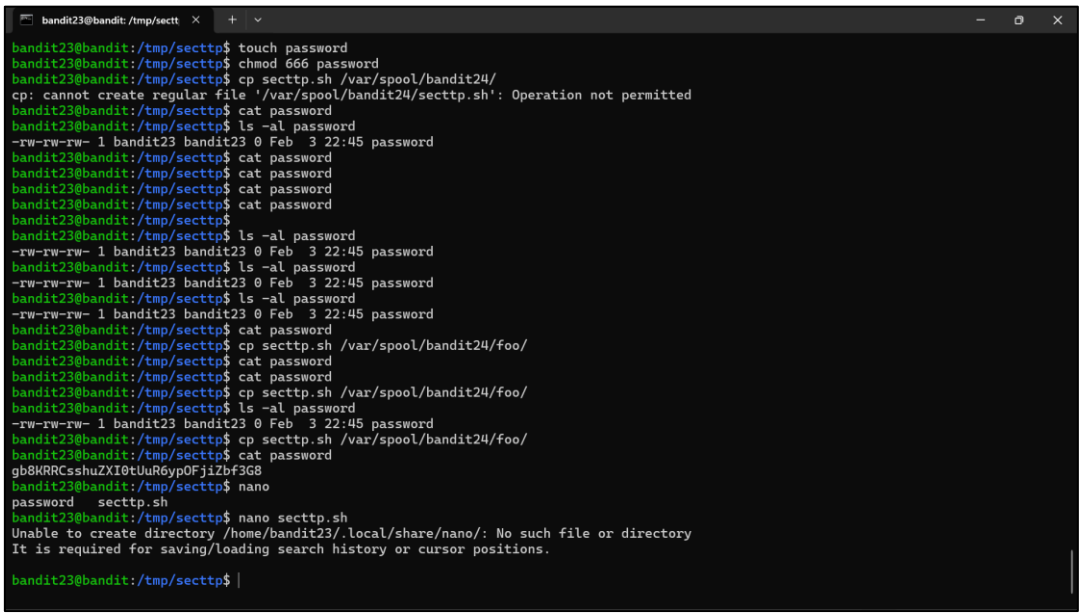




This will take the file and paste the info that has on it in the file that we choose.

cp secttp.sh /var/spool/bandit24/foo: Copies secttp.sh to /var/spool/bandit24/ with the name foo.

touch password: Creates an empty file named password in the /tmp/secttp directory.



Copying the Script to the Cron Job Directory Move the script to the directory where the cron job will pick it up. The following command is used 'cp secttp.sh /var/spool/bandit24/foo'

After the cron job executes the script, check the contents of the password file 'cat password' and at the end the file will find the password.

Result: **gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8**

Level 24 → Level 25

Objective

The objective of this level is to find the password by providing the password of the bandit 24 and a secret numeric 4-digit pin code. This only by going to all the 10000 combinations by brute-forcing

Methodology

We are going to cerate a directory in which the script and the other documents will be stored with the following commands: '*mkdir /tmp/pass24*' and '*cd /tmp/pass24*'

```
bandit24@bandit:~$ cd /tmp/pass24
bandit24@bandit:/tmp/pass24$ ls
bruteforce.sh  pins.txt
bandit24@bandit:/tmp/pass24$ nano bruteforce.sh
Unable to create directory /home/bandit24/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.

bandit24@bandit:/tmp/pass24$ chmod +x bruteforce.sh
bandit24@bandit:/tmp/pass24$ ./bruteforce.sh
bandit24@bandit:/tmp/pass24$
```

Then we use bruteforce in order to identify the correct code. This will drop it in the archive pins.txt

```
bandit24@bandit: /tmp/pass24
  GNU nano 7.2
#!/bin/bash
for i in {000..9999}
do
        echo "gb8KRRCsshuZXI0tUuR6ypOFjiZbf3G8 $i" >> pins.txt
done
```

This is the shell that we will execute

```
bandit24@bandit:/tmp/pass24$ chmod +x bruteforce.sh
bandit24@bandit:/tmp/pass24$ ./bruteforce.sh
bandit24@bandit:/tmp/pass24$ nc localhost 30002 < pins.txt
```

Then we execute the shell

nc localhost 30002 < pins.txt: Sends the contents of pins.txt to the daemon on port 30002. The daemon will process the file and return the password for bandit25 if the correct PIN is included.

```
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Wrong! Please enter the correct current password and pincode. Try again.
Correct!
The password of user bandit25 is iCi86ttT4KSNe1armKiwbQNmB3YJP3q4
```

R**esult: iCi86ttT4KSNe1armKiwbQNmB3YJP3q4**

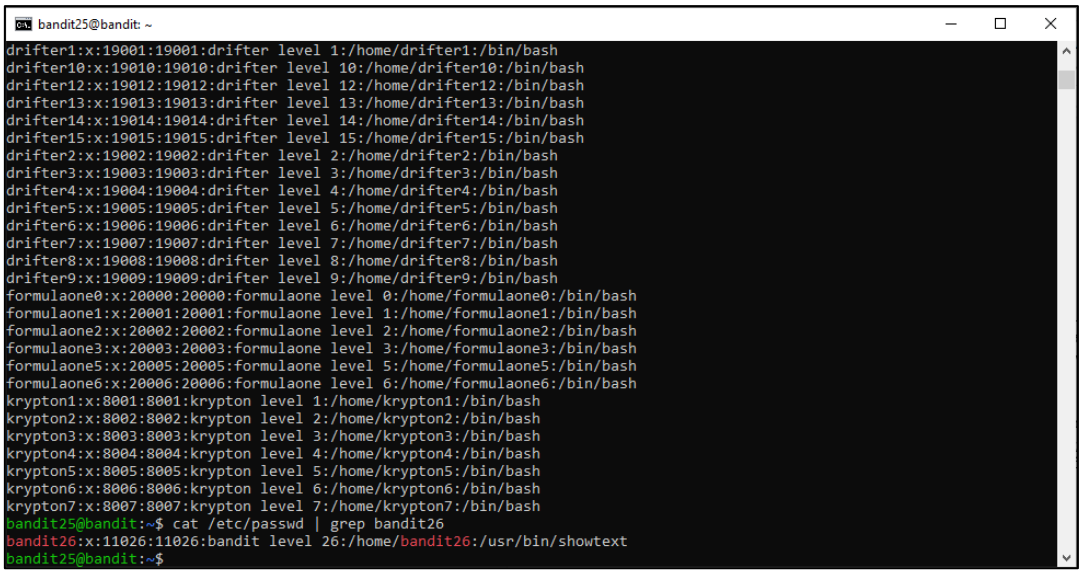
Level 25 → Level 26

Objective

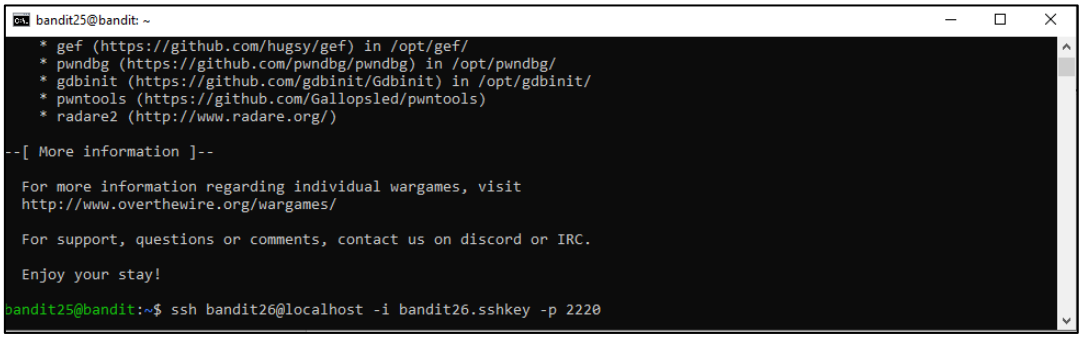
The objective of this level is to find the password by Logging in to bandit26 from bandit25 should be fairly easy… The shell for user bandit26 is not /bin/bash, but something else. Find out what it is, how it works and how to break out of it.

Methodology

We are going to identify the shell for bandit26. In order to determine the shell used by using the following command*: cat /etc/passwd | grep bandit26*. By the usage of this command we will have the correct bash for bandit26.



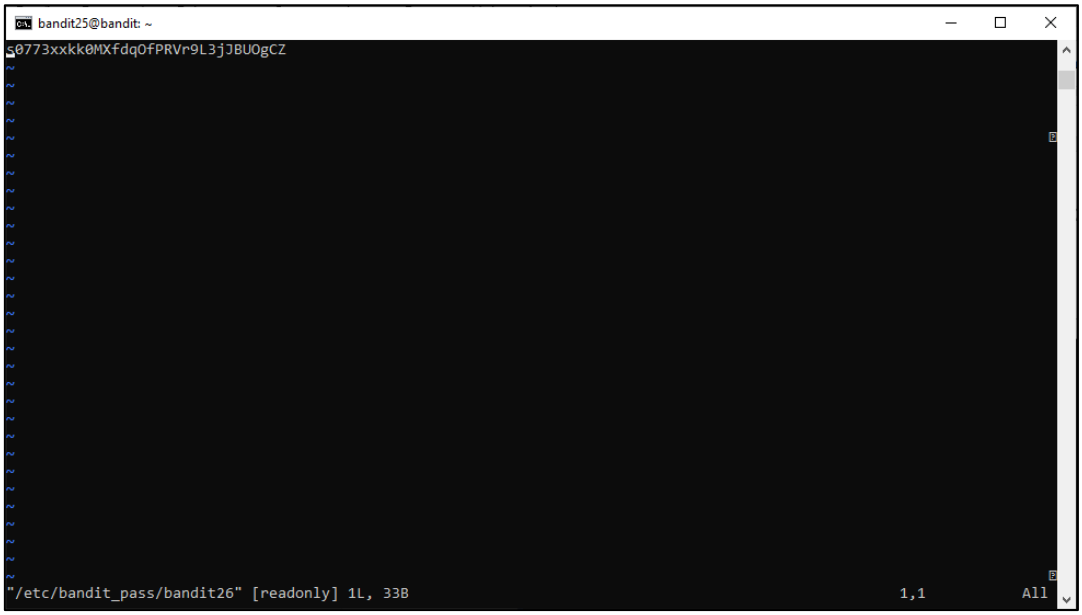
Then we are going to try tod log in into bandit26 in the bash of bandit25. This by using the following command. This command will use the key that is contained in the file bandit26.sshkey



In this part we have to take advantage of a type of bug that allow us to enter in vin at the bash of bandit26 and then we just use a cat command.


bandit25@bandit: ~
1,3
All


bandit25@bandit: ~
:e /etc/bandit_pass/bandit26



Result: s0773xxkk0MXfdqOfPRVr9L3jJBUOgCZ

Level 26 → Level 27

Objective

The objective of this level is to obtain a shell and use it to quickly retrieve the password for the next level, bandit27.

Methodology

For this level, we continue to take advantage of the "bug" in ResultsmoreResults, which allows us to execute commands in ResultsviResults, and through the pass directory, we obtain the password for bandit27.

Result: upsNCc7vzaRDx6oZC6GiR6ERwe1MowGB

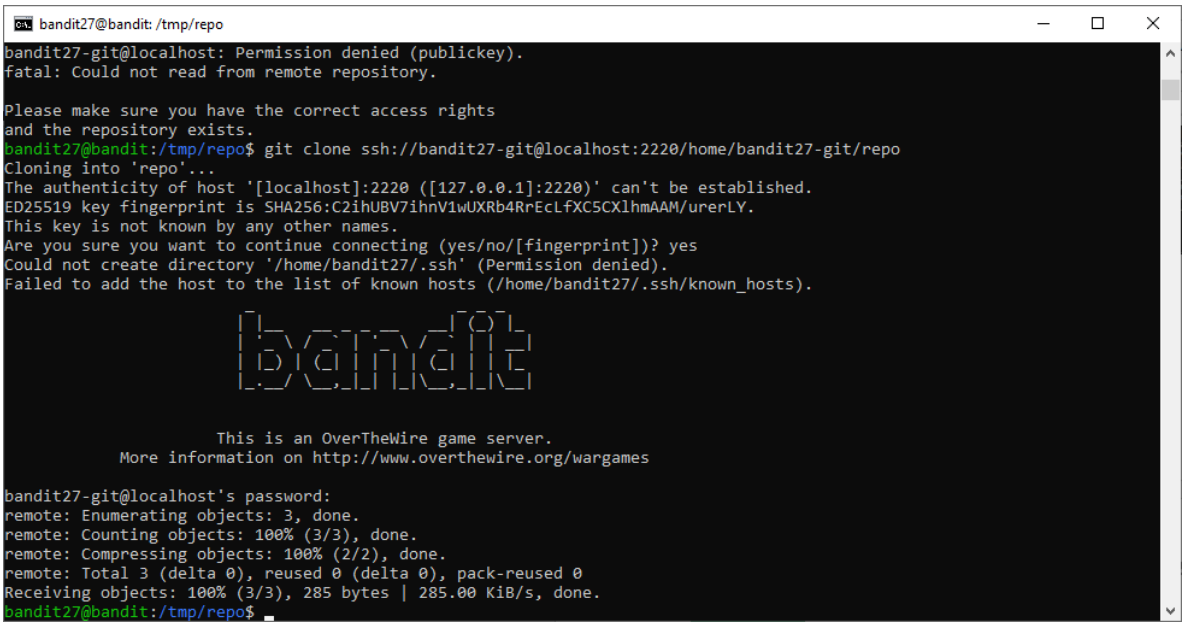
Level 27 → Level 28

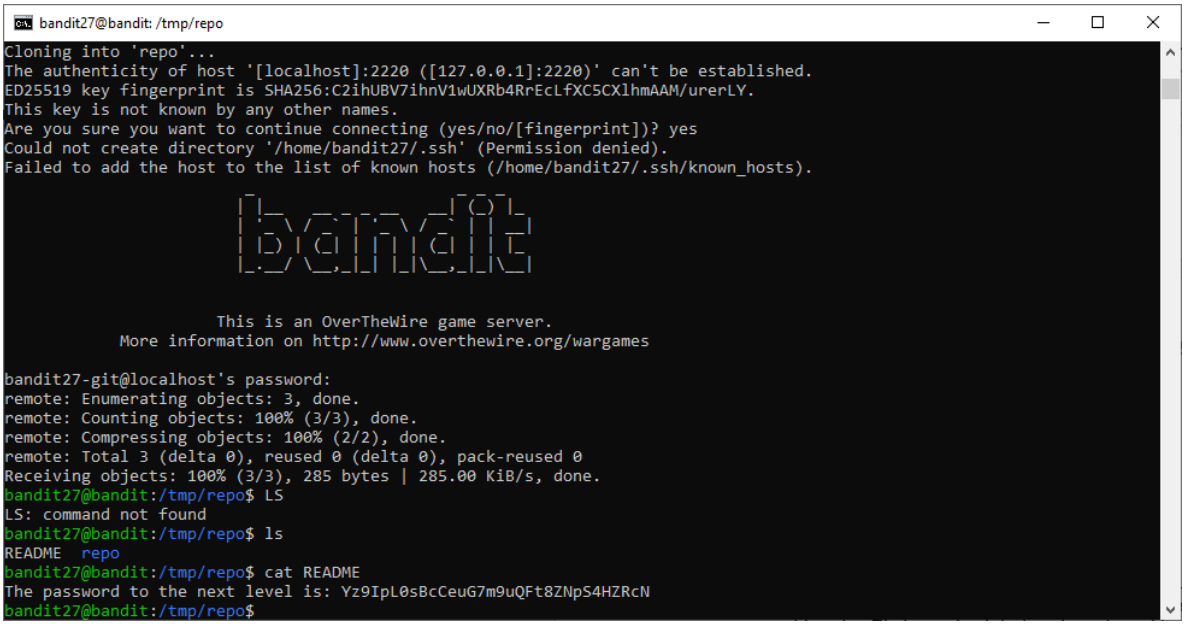
Objective

The objective of this level is to access the git repository at ssh://bandit27-git@localhost/home/bandit27-git/repo via port 2220. The password for the user bandit27-git is the same as the one for the user bandit27.

Methodology

Once cloned, you need to navigate through the repository and look for the Results***readme***Results file, as this file generally contains additional information, and in this specific case, the password for the next level.





Result: Yz9IpL0sBcCeuG7m9uQFt8ZNpS4HZRcN

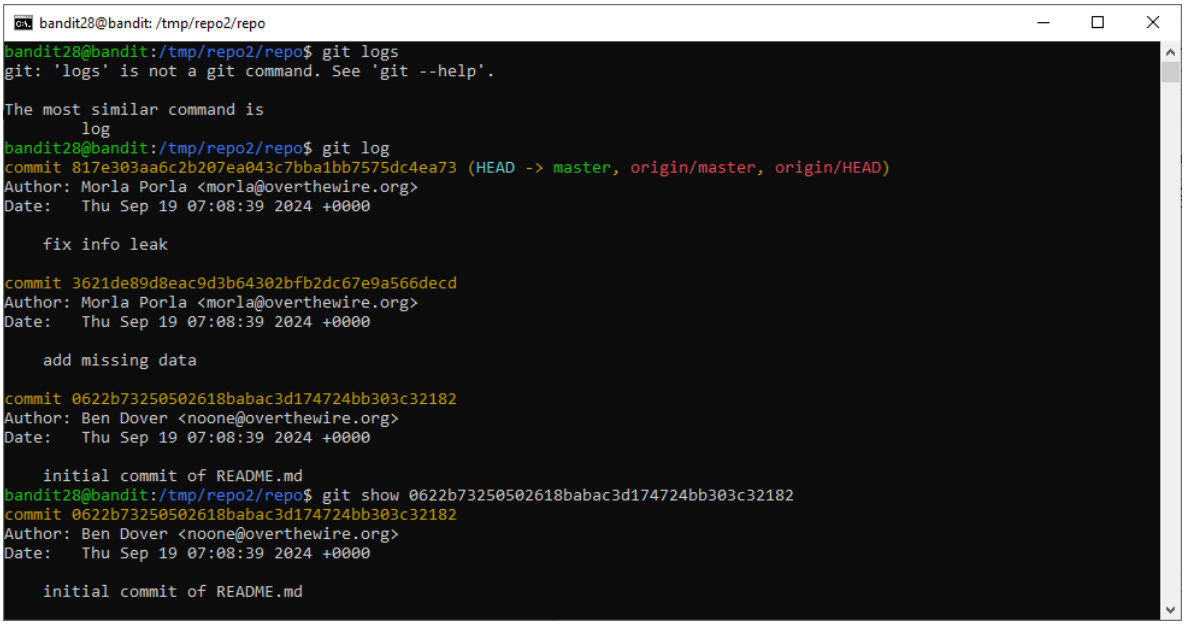
Level 28 → Level 29

Objective

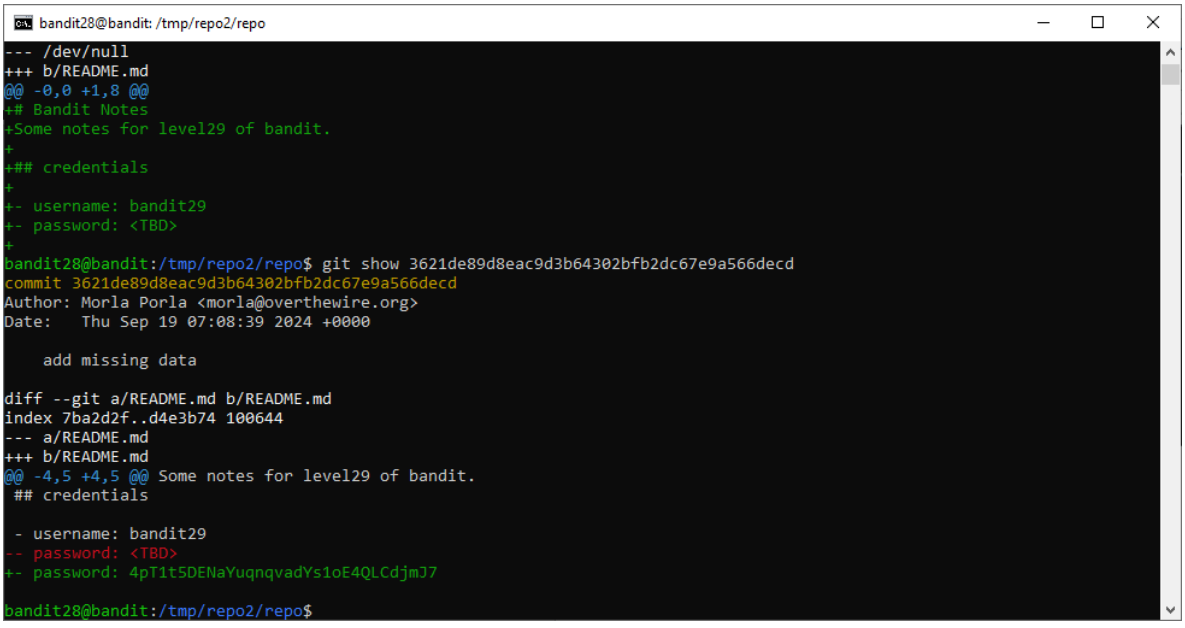
The objective of this level is to access the git repository at ssh://bandit28-git@localhost/home/bandit28-git/repo via port 2220. The password for the user bandit28-git is the same as the one for the user bandit28.

Methodology

First, we create a folder in /tmp to clone the repository and check the file inside. Since that file doesn't provide the key, we use git log to view previous commits and switch to the branch that made the last commit. Finally, we open the readme file, where we find the credentials for the next level.



In this part we use the command show using as a parameter the code of the commit.



And just by trying with two commits we found the password.

R**esult: 4pT1t5DENaYuqnqvadYs1oE4QLCdjmJ7**

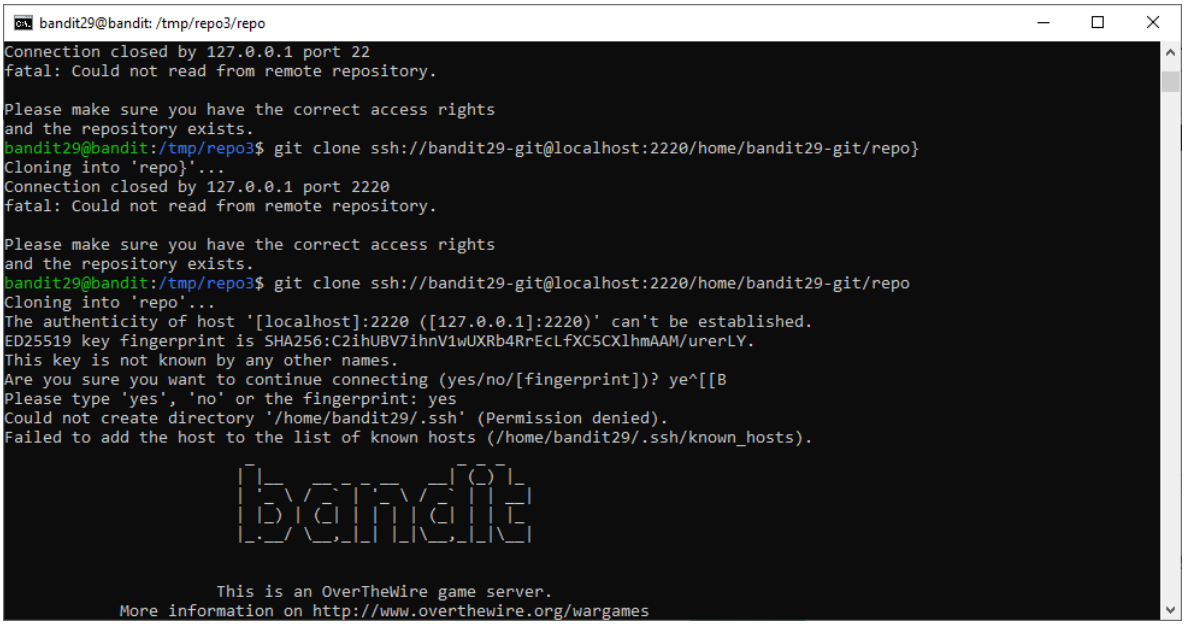
Level 29 → Level 30

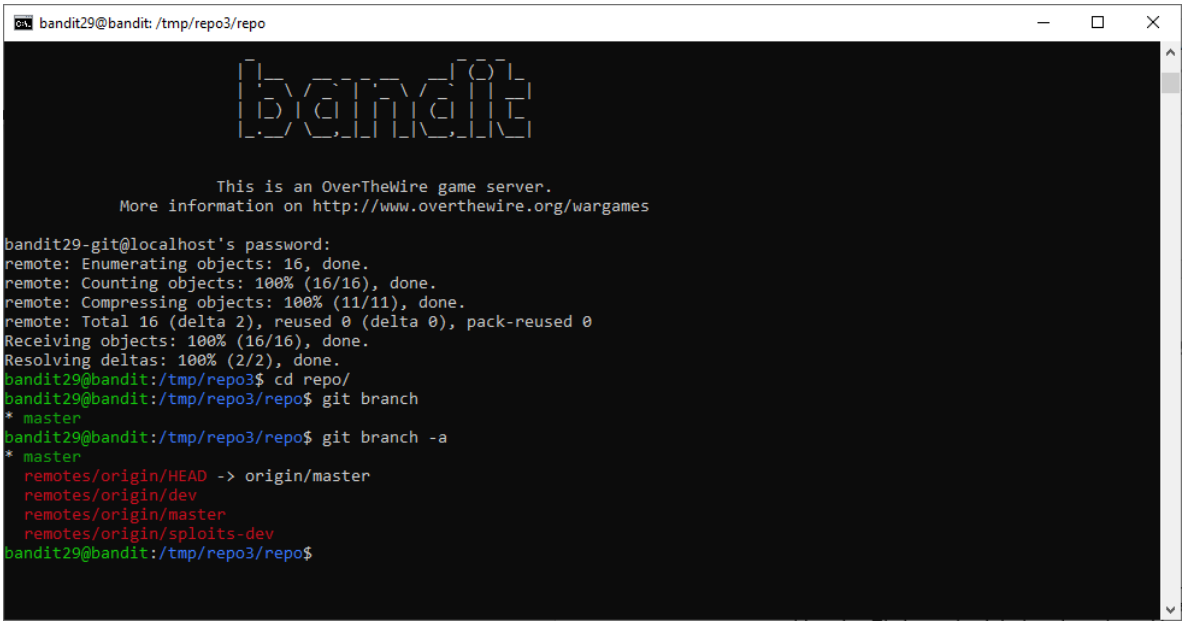
Objective

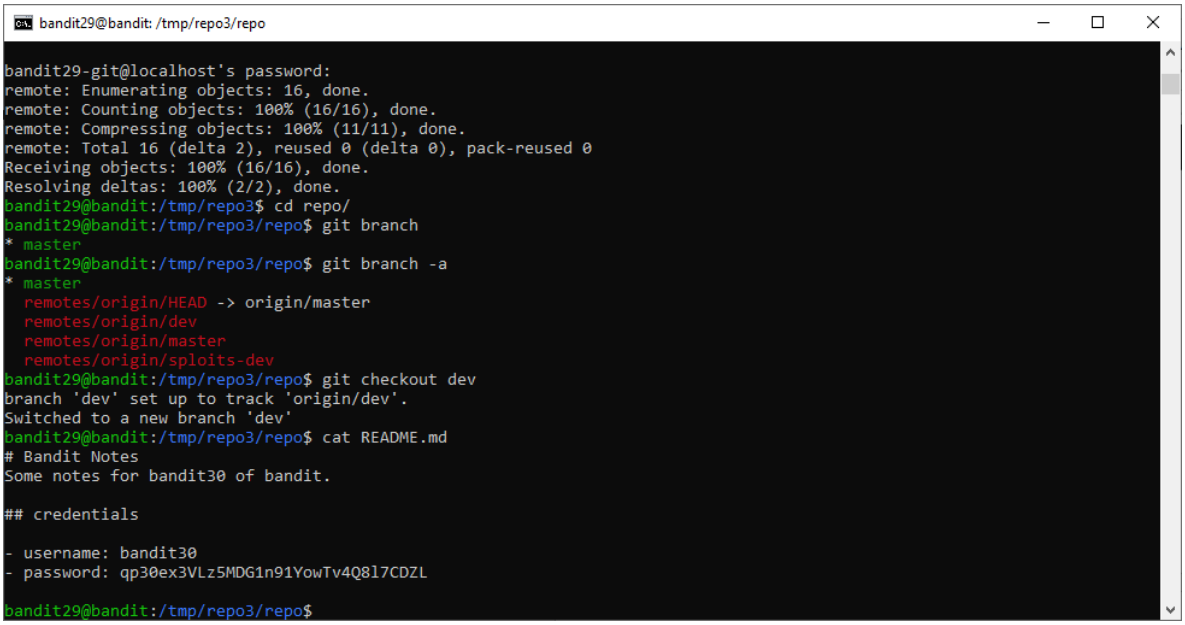
The objective of this level is to clone the git repository at ssh://bandit29-git@localhost/home/bandit29-git/repo via port 2220. The password for the user bandit29-git is the same as the one for the user bandit29. Once cloned, find the password for the next level.

Methodology

For this level, we follow the same steps as the previous one until we reach the readme.txt, which says "no password in production." After that, we check the branches and switch to the dev branch, where we find the key for this level.







Results: qp30ex3VLz5MDG1n91YowTv4Q8l7CDZL

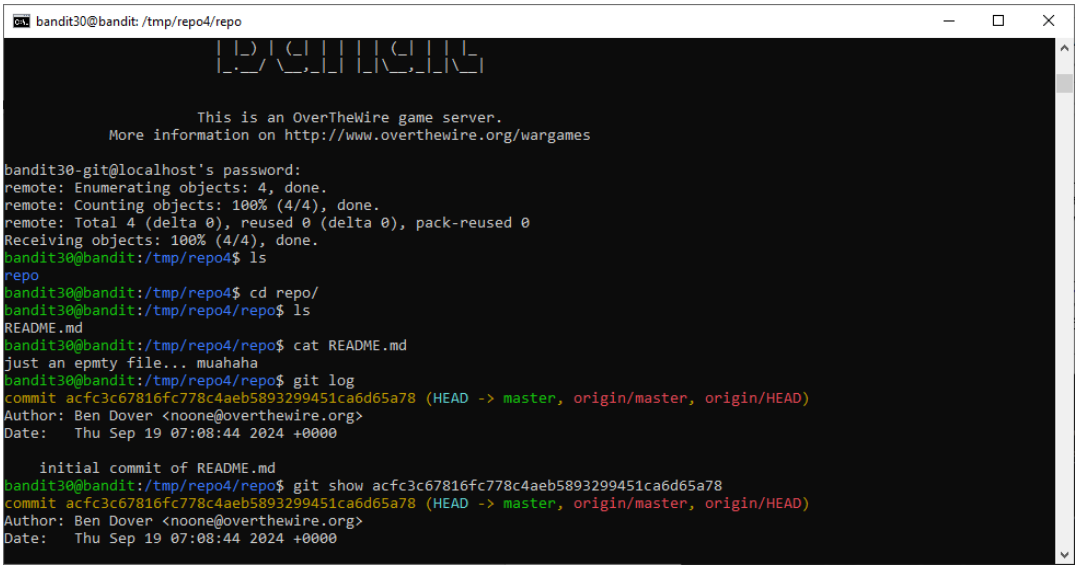
Level 30 → Level 31

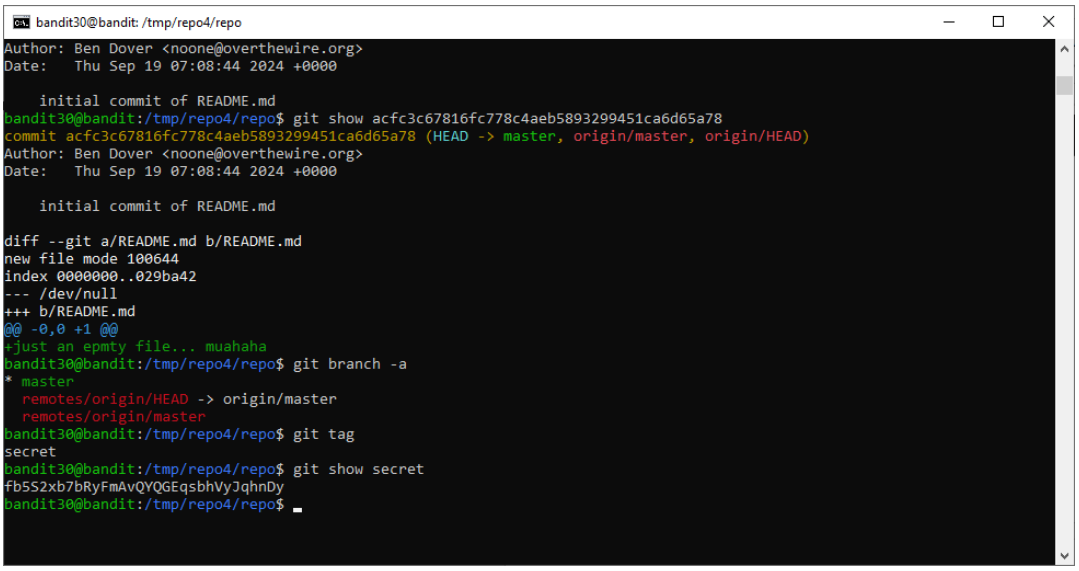
Objective

The objective of this level is to find the password by cloning the repository using the URL provided in the support material of the website. Then we must move in the repo to find clues of the password.

Methodology

First, a temporary directory was created in /tmp to organize the workspace, and the repository was cloned using the provided SSH URL. Then, the cloned repository was navigated to, and the commit history was examined using the git log command to look for relevant information. After identifying a relevant commit, the commit's content was inspected using git show along with the commit hash. Next, the tags in the repository were listed using git tag, and the "secret" tag was examined with git show secret, where the password for the next level was finally found.





- **mkdir -p /tmp/b30**: Creates a temporary directory named b30 in /tmp.
- **cd /tmp/b30**: Navigates into the b30 directory.
- **git clone ssh**://bandit30-git@localhost/home/bandit30-git/repo: Clones the Git repository from the provided SSH URL.
- **cd repo**: Changes into the cloned repository's directory.
- **git log**: Displays the commit history of the repository, including commit messages, authors, and commit hashes.
- **git show numberCommit**: Displays the content and changes associated with a specific commit identified by its hash.
- **git tag:** Lists all tags in the repository, which may point to significant commits or information.

- **git show secret**: Displays the content associated with the "secret" tag, revealing important information.

Result: fb5S2xb7bRyFmAvQYQGEqsbhVyJqhnDy

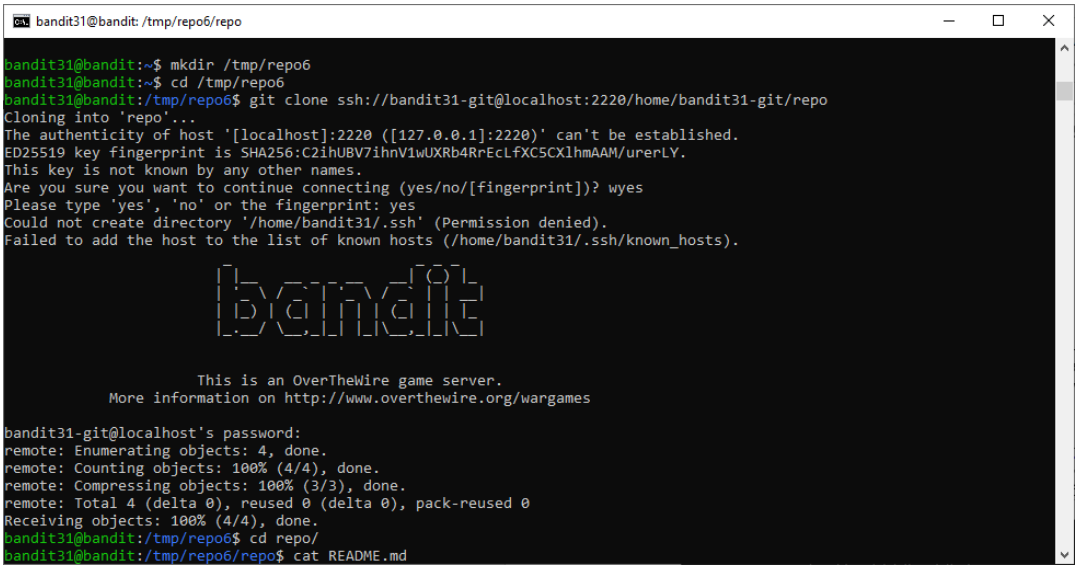
Level 31 → Level 32

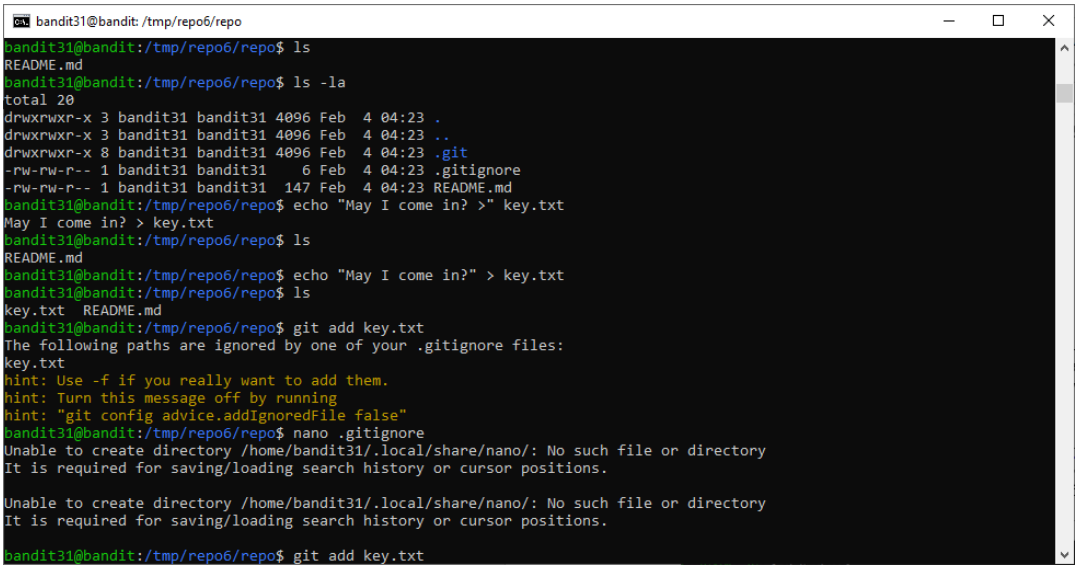
Objective

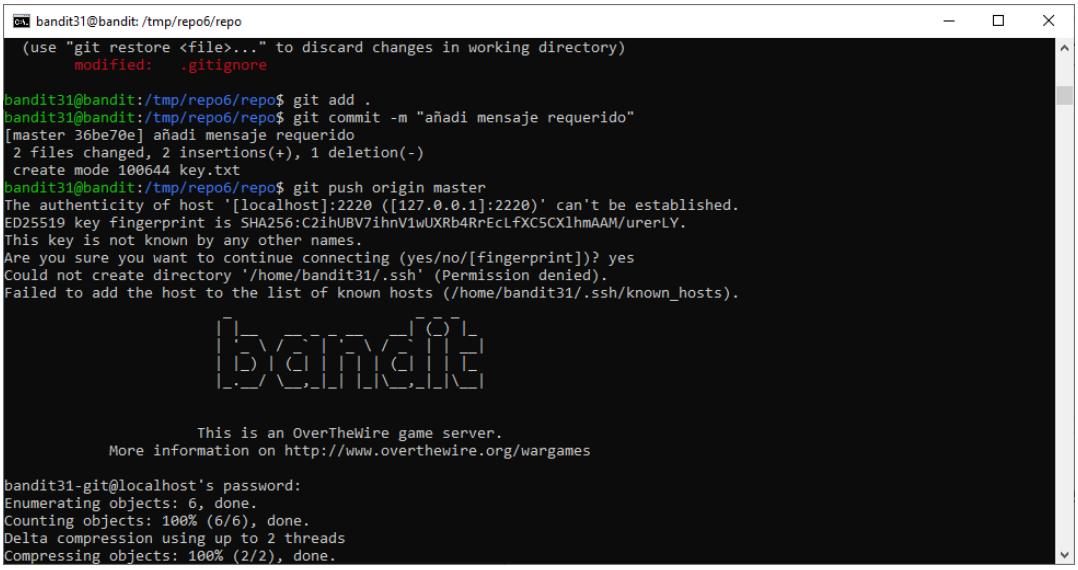
The objective of this level is to find the password by Logging in to bandit26 from bandit25 should be fairly easy… The shell for user bandit26 is not /bin/bash, but something else. Find out what it is, how it works and how to break out of it.

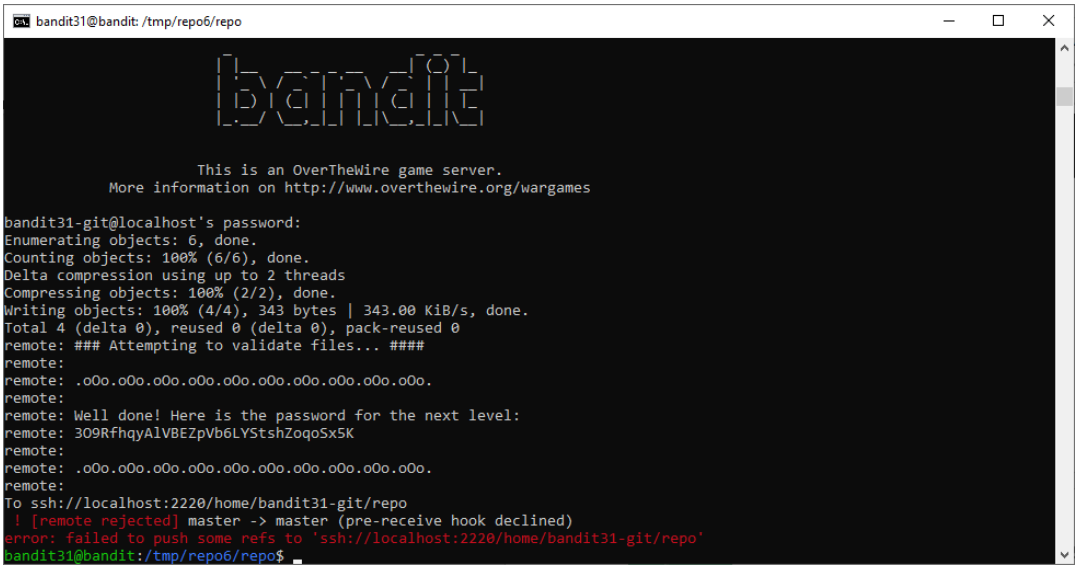
Methodology

To complete this task, first, a temporary directory is created and navigated to using **mkdir -p /tmp/b31** and **cd /tmp/b31**. The Git repository is then cloned using git clone **ssh://bandit31-git@localhost/home/bandit31-git/repo**. After cloning, the README file is read with cat README.md, which provides instructions. Following these instructions, a key.txt file is created with the content "May I come in?" using echo "May I come in?" > key.txt. Next, the **.gitignore** file is edited with vim .gitignore to stop ignoring .txt files. The changes are staged with git add key.txt and committed with **git commit -m "requiredMessage".** Finally, the changes are pushed to the remote repository using git push origin master, successfully revealing the password for the next level.




bandit31@bandit: /tmp/repo6/repo
bandit31@bandit:/tmp/repo6/repo$ ls
README.md
bandit31@bandit:/tmp/repo6/repo$ ls -la
total 20
drwxrwxr-x 3 bandit31 bandit31 4096 Feb 4 04:23 .
drwxrwxr-x 3 bandit31 bandit31 4096 Feb 4 04:23 ..
drwxrwxr-x 8 bandit31 bandit31 4096 Feb 4 04:23 .git
-rw-rw-r-- 1 bandit31 bandit31   6 Feb 4 04:23 .gitignore
-rw-rw-r-- 1 bandit31 bandit31 147 Feb 4 04:23 README.md
bandit31@bandit:/tmp/repo6/repo$ echo "May I come in? >" key.txt
May I come in? > key.txt
bandit31@bandit:/tmp/repo6/repo$ ls
README.md
bandit31@bandit:/tmp/repo6/repo$ echo "May I come in?" > key.txt
bandit31@bandit:/tmp/repo6/repo$ ls
key.txt README.md
bandit31@bandit:/tmp/repo6/repo$ git add key.txt
The following paths are ignored by one of your .gitignore files:
key.txt
hint: Use -f if you really want to add them.
hint: Turn this message off by running
hint: "git config advice.addIgnoredFile false"
bandit31@bandit:/tmp/repo6/repo$ nano .gitignore
Unable to create directory /home/bandit31/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.
Unable to create directory /home/bandit31/.local/share/nano/: No such file or directory
It is required for saving/loading search history or cursor positions.
bandit31@bandit:/tmp/repo6/repo$ git add key.txt


bandit31@bandit: /tmp/repo6/repo
(use "git restore <file>..." to discard changes in working directory)
modified: .gitignore
bandit31@bandit:/tmp/repo6/repo$ git add .
bandit31@bandit:/tmp/repo6/repo$ git commit -m "añadi mensaje requerido"
[master 36be70e] añadi mensaje requerido
2 files changed, 2 insertions(+), 1 deletion(-)
create mode 100644 key.txt
bandit31@bandit:/tmp/repo6/repo$ git push origin master
The authenticity of host '[localhost]:2220 ([127.0.0.1]:2220)' can't be established.
ED25519 key fingerprint is SHA256:C2ihUBV7ihnV1wUXRb4RrEcLfXC5CXlhmAAM/urerLY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Could not create directory '/home/bandit31/.ssh' (Permission denied).
Failed to add the host to the list of known hosts (/home/bandit31/.ssh/known_hosts).
This is an OverTheWire game server.
More information on http://www.overthewire.org/wargames
bandit31-git@localhost's password:
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.



Result: 3O9RfhqyAlVBEZpVb6LYStshZoqoSx5K

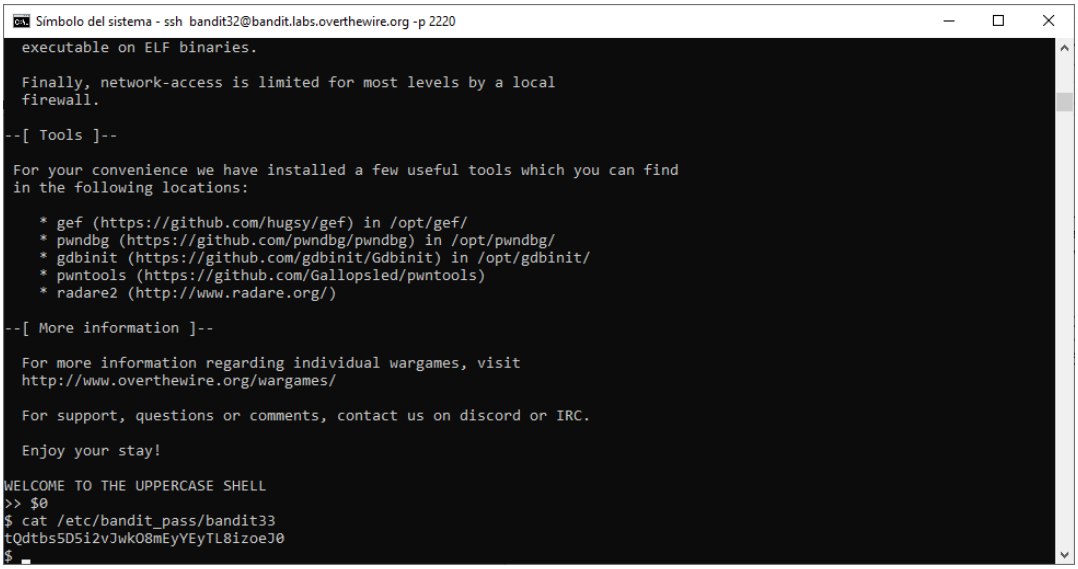
Level 32 → Level 33

Objective

The objective of this level is to find the password by implementing new commands and learn how does other types of bash works.

Methodology

We are going to only use the command $0: In Bash, $0 can be used to print the name of the shell or script. In this context, it reveals information about the shell being used.



This give us the actual password.

Result: tQdtbs5D5i2vJwkO8mEyYEyTL8izoeJ0

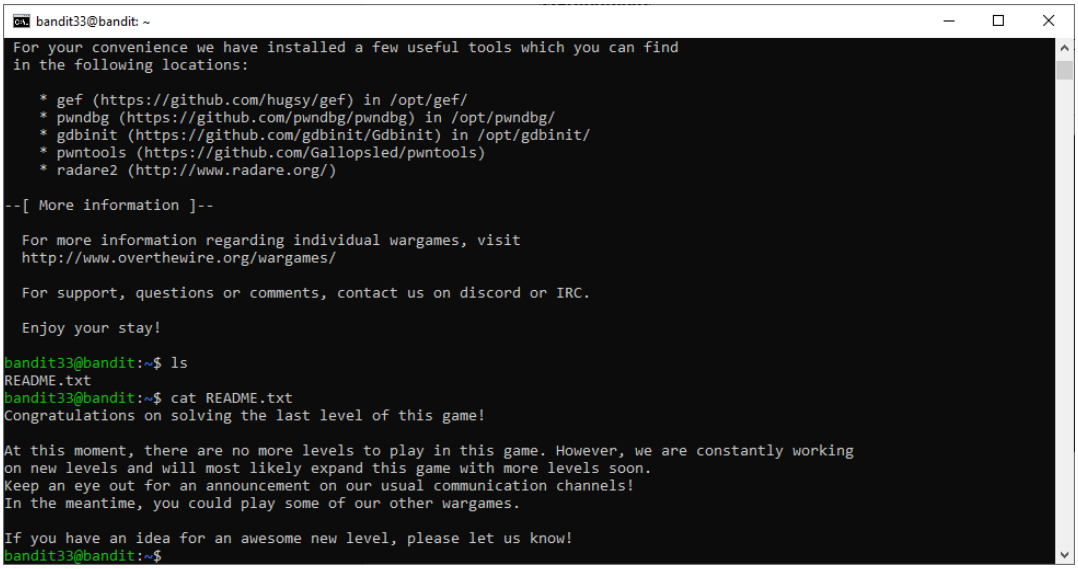
Level 33 → Level 34

Objective

The objective of this level is to find the password.

Methodology

In this level we only have to put the command cat for read the README file and there was the congratulations for finishing the entire bandit program.



Bibliography

Canonical. (s/f-a). *Ubuntu Manpage: cat - concatenate files and print on the standard output*. Ubuntu.com. Recuperado el 4 de febrero de 2025, de https://manpages.ubuntu.com/manpages/noble/man1/cat.1.html

Canonical. (s/f-b). *Ubuntu Manpage: du - estimate file space usage*. Ubuntu.com. Recuperado el 4 de febrero de 2025, de https://manpages.ubuntu.com/manpages/noble/man1/du.1.html

Canonical. (s/f-c). *Ubuntu Manpage: find - search for files in a directory hierarchy*. Ubuntu.com. Recuperado el 4 de febrero de 2025, de https://manpages.ubuntu.com/manpages/noble/man1/find.1.html

Canonical. (s/f-d). *Ubuntu Manpage: ls - list directory contents*. Ubuntu.com. Recuperado el 4 de febrero de 2025, de https://manpages.ubuntu.com/manpages/noble/man1/ls.1.html

Canonical. (s/f-e). *Ubuntu Manpage: This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface*. Ubuntu.com. Recuperado el 4 de febrero de 2025, de https://manpages.ubuntu.com/manpages/noble/man1/cd.1posix.html

Serna, I. (2021, marzo 24). *Tutorial del comando Grep – Cómo buscar un archivo en Linux y Unix con búsqueda recursiva*. freecodecamp.org. https://www.freecodecamp.org/espanol/news/grep-command-tutorial-how-to-search-for-a-file-in-linux-and-unix-with-recursive-find/