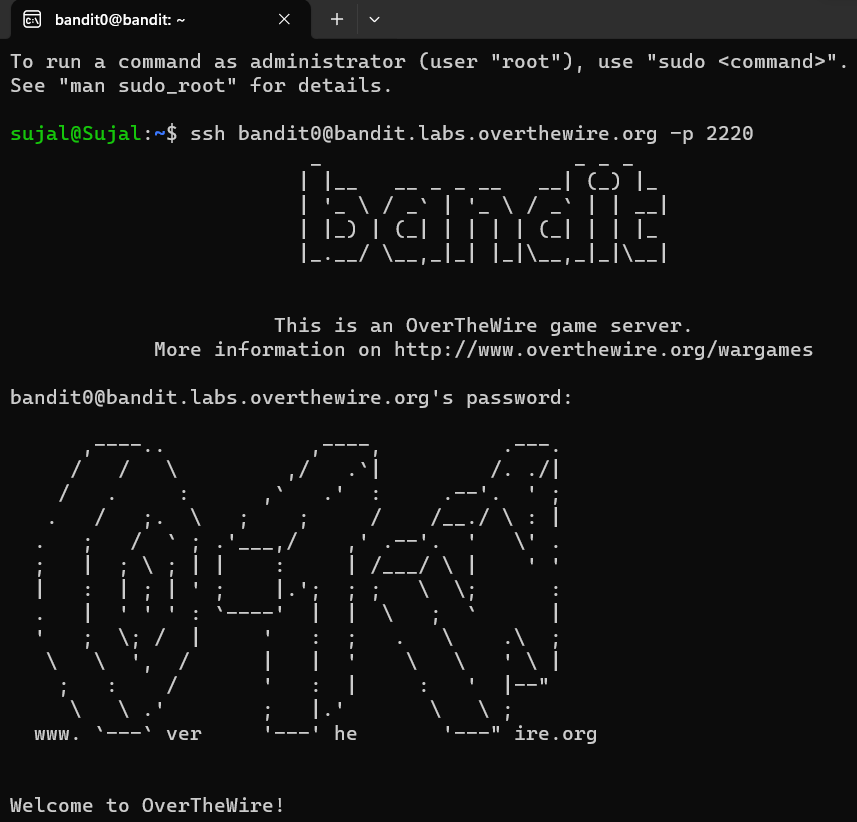
**OVER THE WIRE**

**Bandit Levels Walkthroughs**

**Level 0**

The goal of this level is for me to log into the game using Secure Shell Protocol (SSH), which is used to establish a secure remote connection between the machines. The basic command structure of **SSH** command is, **ssh <username>@<server> -p <port>**. Given, the server is bandit.labs.overthewire.org and the port is *2220*. Also, the username & password to the level is bandit0. I executed my task in the following manner using *Windows Subsystem for Linux* i.e., via *Ubuntu*.



Similarly, the subsequent levels are unlocked by establishing remote connections using ssh and then hunting for passwords.

**Level 0 → Level 1**

In this level, my task is to hunt for the password to the next level (i.e., Level 1), which is stored in a file called **readme** located in the *home directory*. And by using this password, I must 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. The commands which I may require are ls, cd, cat, file, du, find, which have the following specifications.

1. ls – List directory contents.
2. cd – Change directory from current to other.
3. cat - Concatenate files and print on the standard output.
4. file – Determines the file type.
5. du – Estimates file space usage *(Disk Usage)*.
6. find - Searches for files in a directory hierarchy.
7. rm – Removes files or directories.

The above commands specifications can be explored using *“man”* command *(manual page)* by simply typing *“man”* and the *“command name”* next to it as **man ls**. I checked the directory contents using **ls** command and read the contents of the file readme by executing **cat readme** command, to receive the password to the next level. I executed my task in the following manner.



**Level 1 → Level 2**

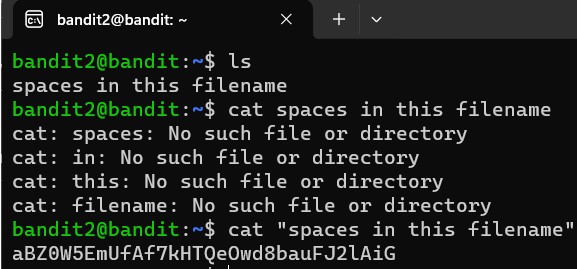
Having received the password for Bandit Level 1, I logged into **bandit1** using **SSH**. Now, my task is to find the password to the next level which is stored in a file called **-** located in the home directory. Initially, like other levels, I list the directory contents using **ls** command and I encountered **‘-’**, which is a special character in Linux and files with this symbol as the first character cannot just be referenced as other files. Just like normal files, it simply cannot be read by using ***cat –*** command directly, rather we need to prefix the file name ***(-)*** with a path **-./-**, or **/home/sujal/-** to make it as readable as other files. This technique is also used to get around similar issues where command line options clash with file names. However, I executed my task in the following manner by executing the command as ***cat ./-***.



**Level 2 → Level 3**

Having received the password for Bandit Level 2, I logged into **bandit2** using **SSH**. Now, my task is to hunt for the password to the next level which is stored in a **file** called **spaces in this filename** located in the home directory.

It’s never a good practice or is unconventional to include spaces in the filename rather underscore (\_) or a dash (-) can be used. It is because, in a command, spaces indicate an addition (files, constraints, etc.) to the command. In this case, the filename contains space and so to read its contents it can be written in the clubbed format using single or double quotes as “**spaces in this filename”** and further it can be executed using cat command as follows.

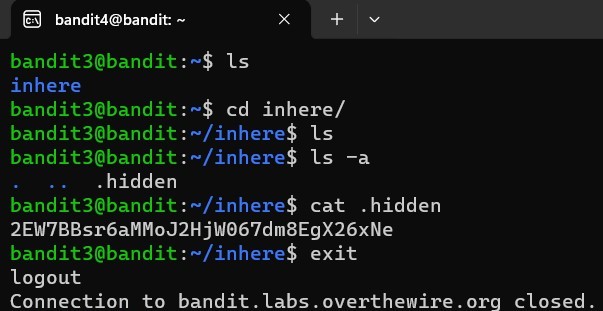


**Level 3 → Level 4**

Having received the password for Bandit Level 3, I logged into **bandit3** using **SSH**. Now, my task is to hunt for the password for the next level that is stored in a *hidden file* in the **inhere** directory. To traverse through directories, **cd** *(change directory)* command is used. It also has the following specifications (path can be relative or absolute).

1. **cd ..** goes to the parent directory.
2. **cd /** goes to the root directory.
3. **cd ~** goes to the home directory *(of the current user)*.

Here, the file is hidden in inhere directory and ls command won’t work simply because it only reflects the unhidden files. A hidden file in Linux starts with a ‘**.’**. To access the hidden files, **-a** *flag* is used along with **ls** command and further **cat** command can be used to read its content. I executed my task as follows.



**Level 4 → Level 5**

Having received the password for Bandit Level 4, I logged into **bandit4** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in the only human-readable file in the inhere directory.

After traversing and listing the contents of the inhere directory by using cd & ls command respectively, I got ***10 files*** *(filenames starting with a dash)* in the terminal, after which I started reading the contents of the files that gave *me junk or garbage or non-human readable values*. Doing this for multiple files is a tedious process and is never advisable for a greater number of files.

A screenshot of a computer program

Description automatically generated

The **file** command gives us the type of data of the file. Data encodings that are human readable are *ASCII a*nd *Unicode*. To implement a command on all the files, a wildcard “**\***” is used along with the command. Finally, I received all the files with their types and hence I got my human readable file i.e., **-file07** containing ASCII text, which I read using **cat** command. I executed my task as follows.

A computer screen with white text

Description automatically generated

**Level 5 → Level 6**

Having received the password for Bandit Level 5, I logged into **bandit5** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in a file somewhere under the inhere directory and has all the properties as human-readable, 1033 bytes in size and is not executable.

While traversing the inhere directory, I listed its contents using **file \*** which are also directories. It is a tedious task to traverse all the directories or sub directories and their contents. So, in order to get the file which has the properties listed above, I used the constraints or *flags* as **-readable, -size 1033c and ! -executable** along with the **find** command as follows. Here, 1033c represents 1033 bytes and ‘**!**’ states not operator. Finally, I read the desired file to get my password.

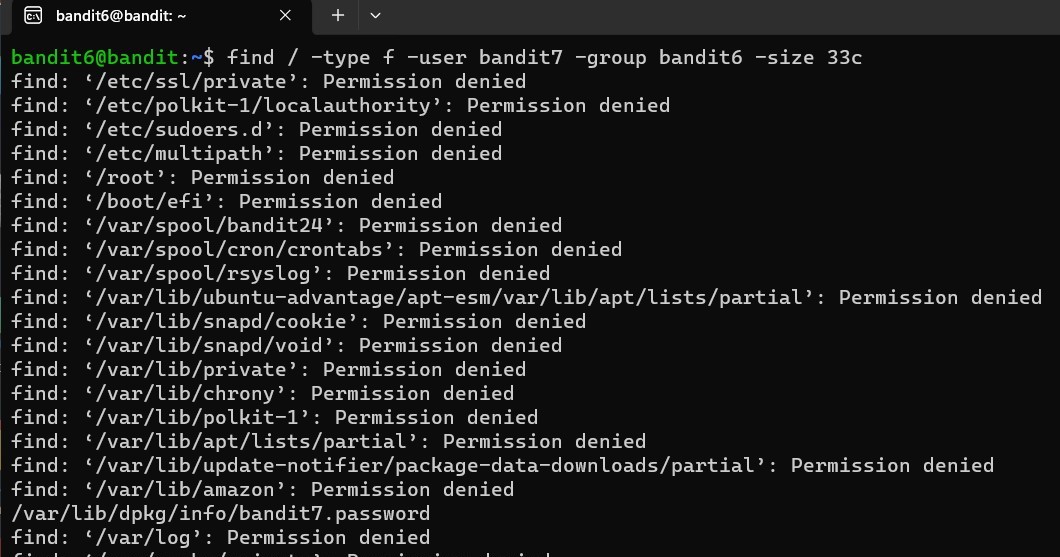
A screen shot of a computer

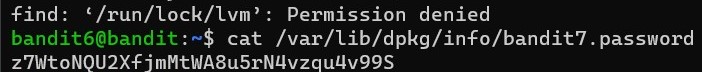
Description automatically generated

**Level 6 → Level 7**

Having received the password for Bandit Level 6, I logged into **bandit6** using **SSH**. Now, my task is to hunt for the password to the next level that is stored somewhere on the server and has all the properties as owned by user bandit7, owned by group bandit6, 33 bytes in size.

For this task I used the **find** command along with flags or constraints as **find / -user bandit7 -group bandit6 -size 33c** *(/ from root folder, -user the owner of the file, -group the group owner of the file, -size the size of the file.).* Finally, I got the file which has access to bandit7 password. I executed my task as follows.





**Level 7 → Level 8**

Having received the password for Bandit Level 7, I logged into **bandit7** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in the file **data.txt** next to the word **millionth**.

For this task, **grep** command can be used which searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. And, with the **pipe** (**|**), we can pipe the output of cat to grep as input to look through a text file. It basically searches the file for a particular pattern. I executed the command **cat data.txt |** **grep “millionth”** as follows to get the desired password.

A screenshot of a computer

Description automatically generated

**Level 8 → Level 9**

Having received the password for Bandit Level 8, I logged into **bandit8** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in the file **data.txt** and is the only line of text that occurs only once.

For this task, **uniq** command can be used (along with *flag* **-u**) to filter adjacent matching lines in the file. It works when the data is sorted, and it can be done by using **sort** command. I executed the command as **sort data.txt | uniq -u** as follows to get the desired password.

A screenshot of a computer

Description automatically generated

**Level 9 → Level 10**

Having received the password for Bandit Level 9, I logged into **bandit9** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in the file **data.txt** in one of the few *human-readable* strings, preceded by several ‘**=**’ characters.

For this task, **strings** command can be used to distinguish human readable text from file data.txt. Thereafter, **grep** command can be used to search for the strings which has several ‘=’ preceded before the password. For this, we can *pipe* (|) the output of ***cat*** to ***strings*** as input to find distinguish human readable text and similarly piping again its output with **grep** *(^= represents several equal to)* command to look for the desired string value. I executed the command as **cat data.txt | strings | grep ^=** as follows to get the desired password.

A screenshot of a computer screen

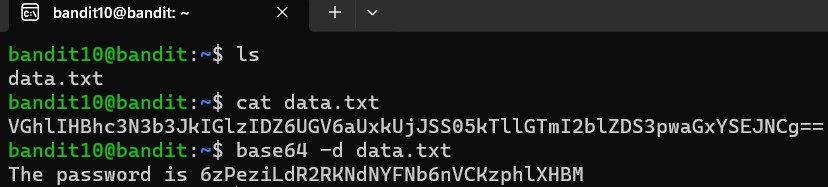
Description automatically generated

**Level 10 → Level 11**

Having received the password for Bandit Level 10, I logged into **bandit10** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in the file data.txt, which contains *base64* encoded data.

In Linux, **base64** is a command that allows to encode or decode in base64. Generally, base64 is a binary-to-text encoding scheme that can often be recognized by equal signs at the end of the data. For this task, after using *cat* command, we can directly execute the command **base64 -d data.txt** or alternatively, we can *pipe* (|) the output of cat to **base64** as input along with *flag* **-d** to decode the encoded data. I executed my task by using the command **cat data.txt | base64 -d** which isas follows.

A screenshot of a computer

Description automatically generated

**Level 11 → Level 12**

Having received the password for Bandit Level 11, I logged into **bandit11** using **SSH**. Now, my task is to hunt for the password to the next level that is stored in the file data.txt, where all lowercase (a-z) and uppercase (A-Z) letters have been rotated by 13 positions.

The file **data.txt** contains a line which is encrypted using **ROT13**, *("rotate by 13 places", sometimes hyphenated ROT-13)* is a simple letter ***substitution cipher*** that replaces a letter with the 13th letter after it in the Latin alphabet. Unfortunately, there is no built-in *ROT13* command in Linux. For that reason, Linux **tr** command, which stands for *‘translate’*, allows replacing characters with others. The base command syntax looks like the following *tr <old\_chars> <new\_chars>*. I executed my task by using the command **cat data.txt | tr ‘a-zA-Z’ ‘n-za-mN-ZA-M’** which isas follows.

