**Overthewire bandit games: Key learnings in the toughest levels:**

**Level 12->13: multiple compressions and hexdump**

Create temp directory using mktemp -d and use cp command to copy the data.txt file into the temp directory

Utilise file command to understand what the file type

Use xxd -r command uncompress hexdump file

Use mv command to rename files to .gz/.bz2/.tar in order to decompress the gz, bz2, tar file types.

Decompress the files using the commands:

* Gzip -d to decompress .gz files
* Bzip2 -d to decompress .bz2 files

Most useful command for me in this level: file - without file, knowing which file was .gz/.bz2/.tar would've been more challenging and time consuming.

**Level 13->14: Using private keys**

This level introduces the use of private keys.

* First use ls to find out our directories and files
* We find “sshkey.private”
* Use the command: file sshkey.private
* We can see that this is a RSA private key
* We can use ssh -i to connect to bandit14 at port 2220

Final command used to connect:

ssh -i sshkey.private bandit14@localhost -p 2220

You are now connected to bandit 14. We need to now find the password, which is located in the /etc/bandit\_pass/bandit14. Use the command ‘cd’ to enter the etc/bandit\_pass directory. Then use ‘cat’ to concatenate the bandit14 file. There you find the password for level 14.

**Level 14-> 15: netcat (nc)**

Introduction to netcat. Netcat is a powerful command that has many uses. In this particular example, it allows us to read/write data to a localhost. We connect to a local host using nc <localhost> <port number>. Then we submit the password for level 14.

**Level 15->16: OpenSSL & s\_client**

This level introduces the use of SSL encryption and using the openssl command. OpenSSL is used to set up secure connections to a localhost using SSL/TLS certifications. The level doesn't ask to setup this connection but to connect to a localhost using SSL/TLS encryption.

First I checked the manpage of openSSL. I stumbled upon s\_client which is a standard command from openSSL that implements a generic SSL/TLS client that establishes a connection to a remote server that speaks SSL/TLS. Now, how exactly do you use the openSSL s\_client command?

I first used the -help command and found many different options to use. I ultimately landed on using -connect option to the localhost on port 30001. The final command line used is:

openssl s\_client -connect localhost:30001

The password of the current level is submitted after entering that command where you are then given the password to the next level.

**Level 16->17: netcat & openSSL**

This challenge tested us on the use of using commands to find out if ports were listening and to see of those listening ports, which of them had SSL/TLS encryption. To do this, first I found out which ports between 31000 and 32000 were open and listening. This can be done using the nc command with the option “-z” (only scans for listening ports without sending data to them).

1st command used:

nc -z bandit.labs.overthewire.org 31000-32000

Open ports listening from the command line above:

* 31960
* 31046
* 31790
* 31691
* 31518

Now we can check which of these ports spoke SSL/TLS. This can be done using the openSSL s\_client -connect command line as we used in previous levels.

I entered each of the ports and found that only 31790 worked. I also had to add -quiet at the end of the command as the password being entered starts with a “k”. This triggers special actions in the linux response, which is why I was initially getting “KEY UPDATE”. Once -quiet was added, this issue was resolved and I was presented with a private key which is used as credentials to login to Bandit17.

Now that we have the RSA private key, we can create a temporary directory where we can store this private key.

Mktemp -d -> cd temp directory -> create file to store private key using touch command -> change permissions to allow both user read and write access using chmod 600 command -> copy the private key text and use vim command to enter the new file, then paste the private key -> now use ssh -i command using localhost to connect to bandit17 (using the private key file you just created). DONE!

**Bandit 17->18: diff command**

Use ‘diff’ command and subtract the lines of text between the two files - you should be left with the new password, since that is the only difference between the two files.

**Bandit 18->19: using ssh -t to bypass .bashrc**

This task involved finding a way to ssh without running the .bashrc directory. To do this we can run the ssh -t option command. First find the list of shells stored. We see that /bin/sh is an option that is not bash related. Combining the -t with /bin/sh allows us to login and access the readme file in the server.

**Bandit 19->20: introduction to setuid**

The level introduces setuid’s at a high-level. When you ls -l, you find a setuid file. This can be identified by the “s” in place of the executable permission for the user (u+s).

* First file the setuid file to confirm it is a setuid file
* Next you can identify what privileges this set user has over the normal user by doing: ./<setuid file name> whoami
  + The answer should be bandit20 - meaning you have privileges and permissions of bandit20
* Next use the command: ./<setuid file name> followed by what action you are trying to complete as the set user. In this example, we are trying to find the password located in the /etc/bandit\_pass/bandit20 file. So, we can concatenate as the setuid to find open the file and retrieve the password. Below is the command used:

./<setuid file name> cat /etc/bandit\_pass/bandit20