HackerFrogs Afterschool
Beginner Cybersecurity Skills

Linux Operation Basics with Bandit CTF: Part 1

Class: Linux OS Operations
Workshop Number: AS-LIN-01
Document Version: 1.0
Special Requirements: None

Table of Contents

Introduction

Welcome to HackerFrogs Afterschool! HackerFrogs Afterschool is a a series of online workshops meant to teach cybersecurity skills and concepts to beginners who may not be familiar with the field of cybersecurity. This lesson is the introductory lesson on Linux OS (operating system) operations, which is an essential skill set, due to the heavy preference of Linux OS in the cybersecurity field, and among ethical hackers in particular. This lesson will cover the following learning objectives:

Learning Objectives:

- Interact with the Linux OS filesystem using basic commands
- Open a Command Line Interface (CLI) terminal on your computer
- Use the SSH program to start an interactive session between your computer and a remote server

What is Linux?

Like Windows and MacOS, Linux is a computer operating system (OS). However, unlike Windows and MacOS, Linux is free, and open-source, which means its source code can be viewed (and potentially modified) by anybody. Linux is a modular OS, which means that software components can be very easily added, updated or removed to fit the user's needs. This has led to the development of different distributions (variations) of Linux that cater to different groups of users. Specifically, cybersecurity-oriented distributions of Linux have been developed, which are ideal for cybersecurity student use.

Why Learn Linux for Cybersecurity?

There are a few important reasons why cybersecurity education favors the use of Linux:

- Linux is easily accessible to students, since it's free

- Many cybersecurity tools / software are available to Linux

- Cybersecurity-oriented distributions of Linux are readily available

Brief Description of Capture The Flag (CTF) Games and Bandit CTF

Capture the Flag (CTF) games are training exercises in the field of cybersecurity. The goal of a CTF exercise is to "capture the flag" through use of cybersecurity skills. In this context, "capture" means to gain access to a file or other resource, and "flag" refers to a secret phrase or password.

The CTF game we will be playing to learn basic Linux skills is called Bandit, which is hosted on the OverTheWire CTF network. The OverTheWire network hosts several different CTF games. The Bandit CTF game is made up of many levels of increasing difficulty. The initial levels (0-5) cover the target Linux commands for this lesson.

Workshop Guide Format Notes

When following instructions in this document, there will be instructions to type specific commands in the CLI window or in a search bar. These commands will be displayed in red, and will have a different `font` applied to them, as well as a light grey background color. For example:

```
this is a sample command text line
```

If there are code snippets in this document, they look similar to the format above, except that the text will be black. For example:

```
this is a sample code snippet line
```

Starting a Command Line Interface Terminal

Before we can access the Bandit CTF game we will need to use a Command Line Interface (CLI) terminal. The following instructions detail how to start a CLI terminal on a Windows 11 system. If you are using a computer running MacOS, please refer to this link (https://www.businessinsider.com/how-to-open-terminal-on-mac), then skip to the next section.

Now let's continue with opening our CLI window.

Step 1

Click on the search bar beside the Windows icon that is located in the middle-left portion of the screen and then type `cmd` in the field, then click on the **Command Prompt** option that appears (see screenshot).
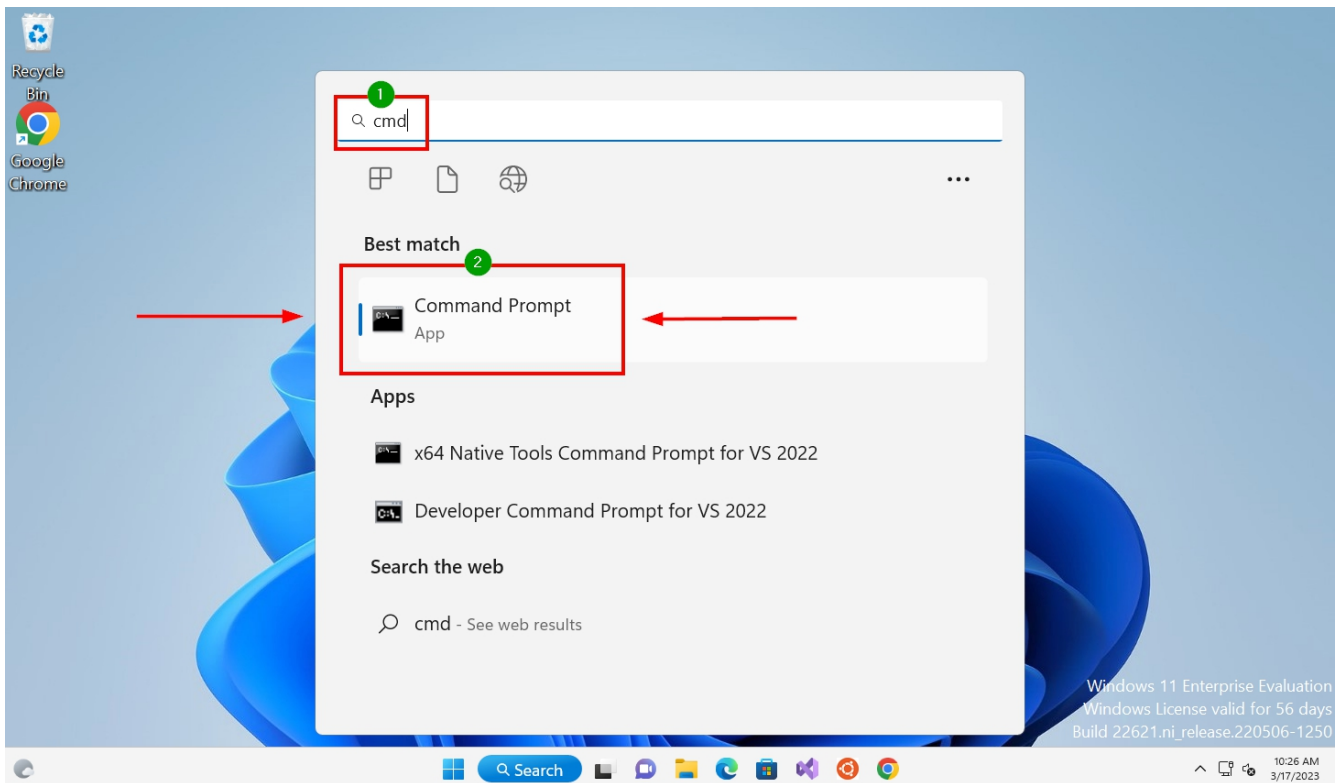
The Command Prompt window that appears is a CLI terminal, and allows users to issues commands to the computer through typed commands.

Step 2 (Only For Windows 10 Users)

If we're using Windows 10, there's one last thing we should do before we use the Command Prompt terminal: adjust one of the properties so we can copy and paste text later. First, click on the icon located at the top-left corner of the **Command Prompt** window, then select **Properties** from the menu that appears (see the screenshot below).
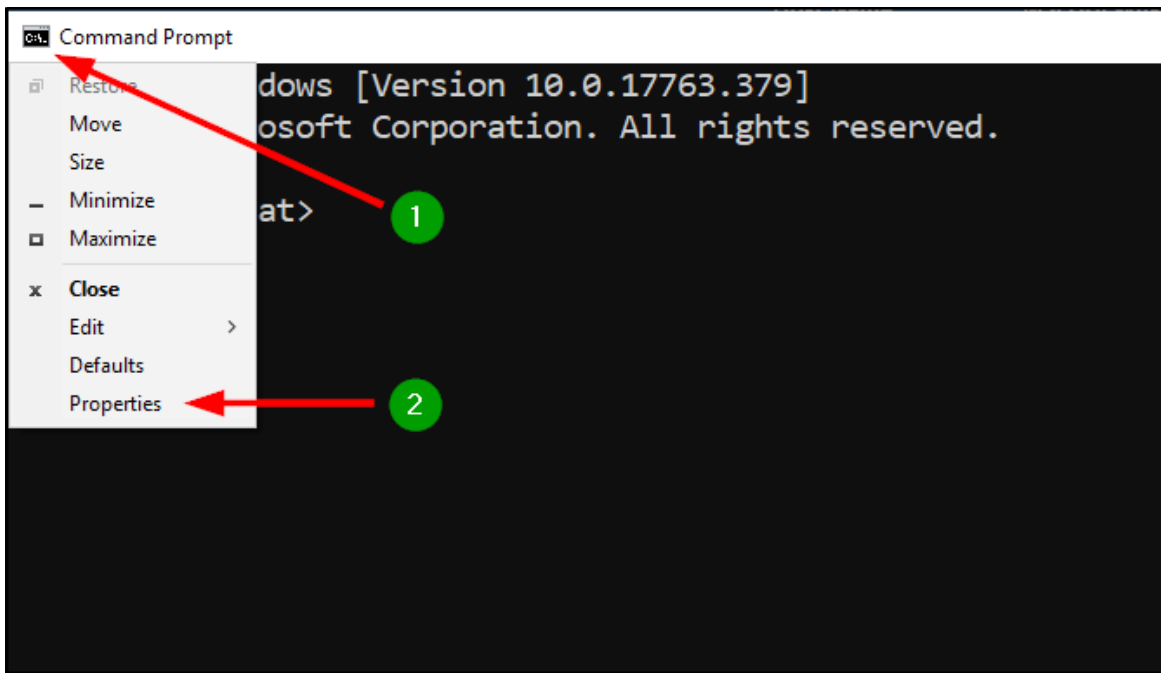
In the **"Command Prompt" Properties** window that appears, click on the **Use CTRL+Shift+C/V as Copy/Paste** checkbox, then click the OK button at the bottom-right portion of the window (see screenshot below).
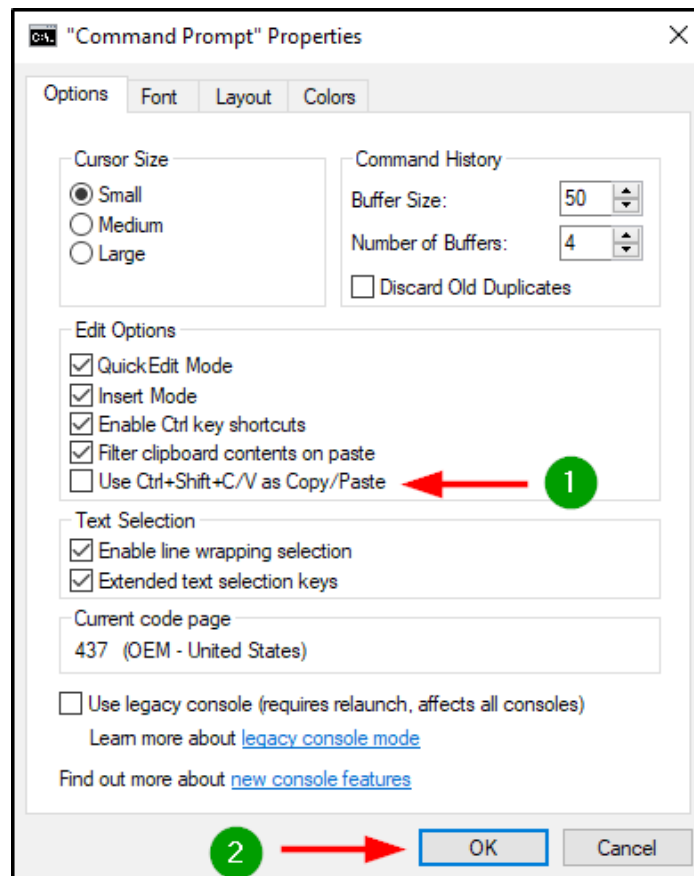
Using the SSH Program to Connect to the CTF Game Server

In the **Command Prompt** window, enter the following command to connect to the the Bandit CTF server using the Secure Shell (**SSH**) program:

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

When prompted if you want to continue connecting, enter **yes**, and when asked for a password, enter **bandit0**. When entering a password into the SSH program, you will not see any response or output while you are typing the password. This is normal. Before entering the password, your screen should look similar to the one below (SSH command and required input outlined in red):

```
C:\Users\shyhat>ssh bandit0@bandit.labs.overthewire.org -p 2220
The authenticity of host '[bandit.labs.overthewire.org]:2220 ([176.9.9.172]:2220)' can't be established.
ECDSA key fingerprint is SHA256:98UL0ZWr85496EtCRkKlo20X3OPnyPSB5tB5RPbhczc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[bandit.labs.overthewire.org]:2220,[176.9.9.172]:2220' (ECDSA) to the list of known hosts.
This is a OverTheWire game server. More information on http://www.overthewire.org/wargames

bandit0@bandit.labs.overthewire.org's password:
Linux bandit.otw.local 5.4.8 x86_64 GNU/Linux
```

Upon a successful login to the Bandit server, you should see a screen similar to the one below:

```
--[ More information ]--

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

  For support, questions or comments, contact us through IRC on
  irc.overthewire.org #wargames.

  Enjoy your stay!

bandit0@bandit:~$
```
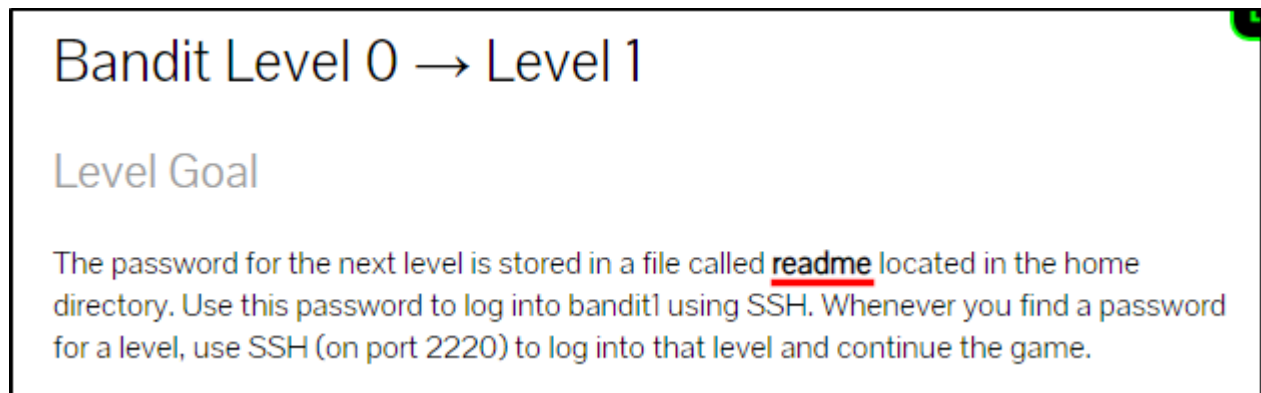
CONTEXT

The SSH program allows users to connect to other SSH servers on the network, i.e., computers that also have the SSH program installed and running in server mode. Once we have connected to the server, any commands that we run in the terminal are done in the context of the user account we've logged in as (in this case bandit0), and we are now interacting with the file system of the SSH server (at overthewire.org). Also, because the SSH server is running Linux, we now have to use Linux commands to interact with it (regardless of whether our own computer is running Windows or MacOS or something else).

Part 0 – Learning the Ls and Cat Commands With Bandit 0

Step 1

First, let's take a look at the Bandit level 0 objectives on the following webpage:

https://overthewire.org/wargames/bandit/bandit1.html

CONTEXT

Each level of the Bandit CTF game has a webpage that describes the objectives of the level. In this case, we are supposed to obtain the flag for this level by reading the file named **readme**. Once we have the flag, we will exit the level and use it as the password to login to the next level as the user **bandit1**.

Step 2

We will list out the contents of the current directory with the following command:

```
ls
```

The resulting output should look like this (the target file is underlined in red):

CONTEXT

The Linux **ls** command lists out of the contents of the current directory if it is executed by itself (without other arguments). Additional features of the **ls** command will be covered in the next levels of the Bandit CTF game.

Step 3

Now that we know the **readme** file we're looking for is in our current directory, we can read its contents with the **cat** command:

`cat readme`

The resulting output will look like this (flag text string underlined in red):

In the screenshot above, the underlined string of letters and numbers is the flag for Bandit level 0.

CONTEXT

Gaining access to flags is the objective of CTF games, and in the Bandit CTF games, the flag for the current level of the game is used as the password for the next level (e.g., the password for Bandit level 1 is the flag for Bandit level 0).

The Linux **cat** command is used to read the contents of files on the server. The **cat** command must be executed with an argument, in this case it's the **readme** file in the current directory.

SPECIAL PASSWORD NOTE: The passwords for the Bandit CTF game change every few months, so all of the passwords that appear in this document are most likely not up to date. Use the passwords found while interacting with the game itself, and document them as they are discovered.

Step 4

The only thing left to do on this level is to logout of the server and record the flag in a file on our own computer. First, enter the following commands to logout from the Bandit CTF server:

`exit`

Then highlight the flag with the mouse cursor in the Command Prompt window then press Ctrl + C to copy the flag. Input the following command, but remember to press **Ctrl + V** to paste in the flag text after **B1**:

`echo B1 <FLAG> >> banditPass.txt`

The output of the previous two commands should look similar to the screenshot below:

```
bandit0@bandit:~$ exit
logout
Connection to bandit.labs.overthewire.org closed.

C:\Users\shyhat>echo B1 boJ9jbbUNNfktd78OOpsqOltutMc3MY1 >> banditPass.txt

C:\Users\shyhat>_
```

CONTEXT

Typically, exiting out of a CLI program (such as SSH) involves entering the command **exit** or **quit**. We also use the **echo** command to identify a text string and include the **>>** operator to save the output to a file called **banditPass.txt**. Now we can recall previous flags (which are used as passwords in the Bandit CTF game) by using the **type** command in the Command Prompt:

```
type banditPass.txt
```

The output will look similar to the screenshot below:

```
C:\Users\shyhat>type banditPass.txt
B1 boJ9jbbUNNfktd78OOpsqOltutMc3MY1
```

Part 1 – Interacting with Special Characters File Names With Bandit 1

Step 1

Login to the Bandit server as the Bandit1 user via SSH:

```
ssh bandit1@bandit.labs.overthewire.org -p2220
```
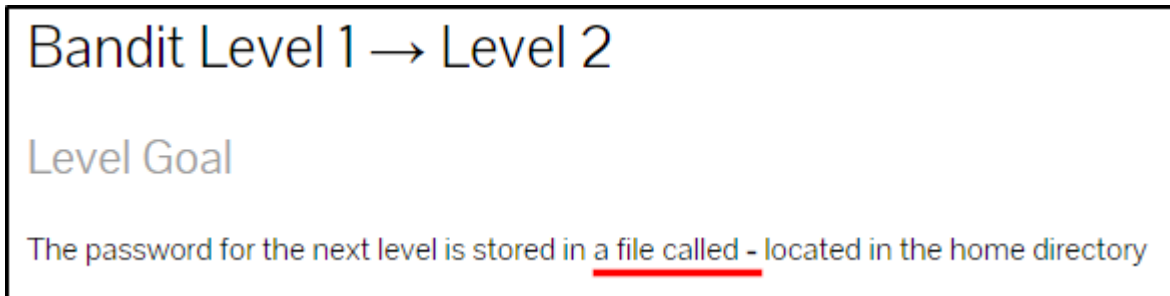password: flag acquired from Bandit level 0

NOTE: When pasting the password into the terminal window you will need to use **Ctrl + Shift + V** instead of just **Ctrl + V,** since the SSH server (overthewire.org) is using Linux.

Also, please note that inside of most CLI terminals, we can access previously entered commands by pressing the **Up Arrow** and **Down Arrow** keys on our keyboard. This feature accesses the terminal's command history, and it can save us a lot of time when using the SSH command because we can use command history to access the previously used SSH command and change just the number portion of the user name we're logging in as (e.g., from bandit0 to bandit1).

Step 2

Look at the level objectives for Bandit level 1 at the following URL:

https://overthewire.org/wargames/bandit/bandit2.html

Bandit Level 1 → Level 2

Level Goal

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

The info on the webpage indicates that the flag for the level is '-', which is a special character in the Linux CLI. We will address how to interact with special characters in file names in the next few steps.

Step 3

List out the contents of the current directory using the **ls** command:

```
ls
```

The output of the command should look like the screenshot below (target file underlined in red):

```
bandit1@bandit:~$ ls
-
bandit1@bandit:~$
```

Now that we know the file is in our current directory, we can use the **cat** command to read the file

```
cat -
```

The output should look like the screenshot below:

```
bandit1@bandit:~$ cat -
```

In this instance, the terminal has become stuck trying to execute this command. Press **Ctrl + C** to interrupt and cancel the current command. Afterwards you should be able to use the terminal again. The output should look like the screenshot below:

CONTEXT

The keyboard shortcut **Ctrl + C** can be used in CLI terminals to stop most commands in the middle of execution. This is very useful if we need to kill a process that's stuck or otherwise unresponsive.

But why did the cat command not execute properly? This is because the system doesn't recognize the '–' character as a file name because the same character is used as a subtraction operator. The system can properly recognize the file if we provide the system with a file path, which is a specific location in the file system. There are a couple of ways to do this, and we'll cover one of these ways in the next step.

Step 4

Let's try reading the file by providing a file path with the following command:

```
cat ./-
```

The output of the command should look like the screenshot below (flag string underlined in red):

CONTEXT

In most Operating System CLI terminals the current directory is indicated by the special character '**.**' (dot or period), and the special character '/' (forward slash) is used when referencing file paths to separate the names of directories and/or files. So **. / –** is interpreted by the system as the file named '-' located in the current directory.

Step 5

Let's finish the level by exiting the server, then recording the flag string to our **banditPass.txt** file:

```
exit
echo B2 <FLAG> >> banditPass.txt
```

The result of the output will look similar to the screenshot below:

```
bandit1@bandit:~$ exit
logout
Connection to bandit.labs.overthewire.org closed.

c:\Users\shyhat>echo B2 CV1DtqXWVFXTvM2F0k09SHz0YwRINYA9 >> banditPass.txt

c:\Users\shyhat>_
```

Part 2 – Interacting with Spaces in File Names With Bandit 2

Step 1

Login to level 2 of the Bandit CTF game with the following command:

```
ssh bandit2@bandit.labs.overthewire.org -p2220
```
password: acquired from Bandit level 1

Step 2

Look at the Bandit level 2 objectives at the following webpage:

https://overthewire.org/wargames/bandit/bandit3.html

Bandit Level 2 → Level 3

Level Goal

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

CONTEXT

The purpose of this level is to teach how to interact with files that have spaces in their file names (which are quite common in files on Microsoft Windows systems). We'll cover two ways to interact with them in the next few steps.

Step 3

List out the current directory's contents with the **ls** command:

```
ls
```

The output should look like the output in the screenshot below (the target file is underlined in red):

Step 4

Let's see what happens when we try using the **cat** command to read the file with all the spaces included:

```
cat spaces in this filename
```


CONTEXT

The **cat** command allows for multiple files to be read at the same time by providing a space in-between the different files, and it appears that when we ran the cat command, the error message indicates that the system was trying to read four separate files with the names "spaces", "in", "this", "and "filename".

Step 5

Let's input the following command to force the system to interpret the spaces in the filename correctly:

```
cat spaces\ in\ this\ filename
```

The output should look like the screenshot below (flag string underlined in red):

CONTEXT

If there are spaces in the names of files we want to interact with (e.g., using the **cat** command), we can force the system to interpret the spaces literally by including the '\' (backslash) escape character directly before the space. Use of the escape character can also be used to literally interpret other special characters as well, such as '$' (dollar sign), and '#' (number sign / hash sign).

Step 6

Let's use the following command as an alternative for interacting with files containing spaces in their names:

```
cat "spaces in this filename"
```

The output should look like the screenshot below (flag string underlined in red):

CONTEXT

Using quotes to encapsulate the name of the file is another way we can interact with files with spaces in their names.

Step 7

To finish this level, we should exit the server and copy the flag to our passwords file:

```
exit
echo B3 <FLAG> >> banditPass.txt
```

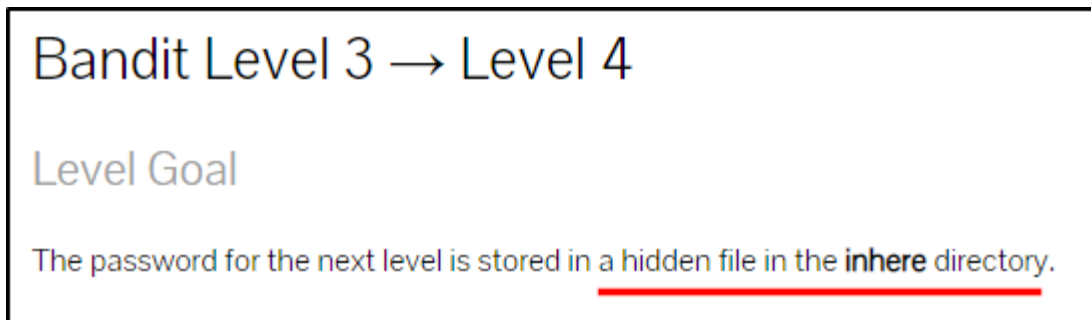# Part 3 – Interacting with Hidden Files With Bandit 3

Step 1

As usual, we start by logging into the Bandit server using SSH:

`ssh bandit3@bandit.labs.overthewire.org -p 2220`
password: flag obtained in Bandit level 2

Step 2

Then we look at the objectives for the level on associated webpage:

https://overthewire.org/wargames/bandit/bandit4.html

In this level, we will will be interacting with files in different directories for the first time, as well as interacting with hidden files.

Step 3

Let's look for the **inhere** directory with the **ls** command:

`ls`

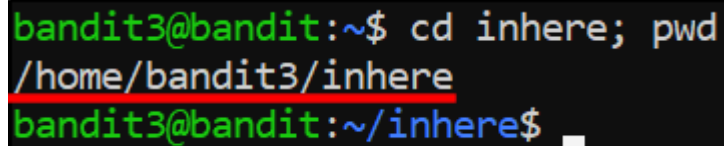The output for this should look like the screenshot below (target directory underlined in red):

Step 4

Next, we should change our current directory by using the **cd** command, then check that we're in the inhere directory with the **pwd** command. We can run both commands one after another with the following command:

```
cd inhere; pwd
```

The output of the previous command is shown in the screenshot below (current directory is underlined in red):

CONTEXT

The **cd** (change directory) command allows us to change the current directory in our terminal session. When using SSH, the default directory after logging in is the user's home directory. E.g., the bandit3 user's home directory is **/home/bandit3**. After the we move into the **inhere** directory with the **cd** command the current directory is then **/home/bandit3/inhere**.

At the end of the **cd** command, we use the **';'** (semicolon) special character to indicate the end of one command. A second command can be included after the semicolon, which is the **pwd** command.
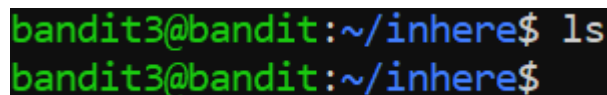
The **pwd** command outputs the current directory for our terminal session. **Pwd** stands for "print working directory".

Step 5

Let's look for the hidden file in the **inhere** directory with the **ls** command:

```
ls
```

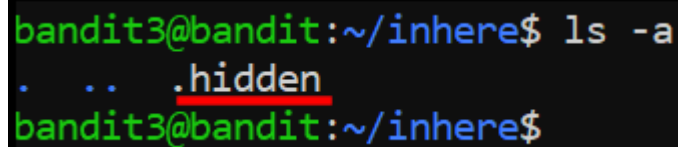The output of this command should look like the following screenshot:

CONTEXT

Regular use of the **ls** command does not reveal any hidden files in directories, so there is no output here.

> ### Step 6

Use the following modified **ls** command to list out hidden files in the current directory:

```
ls -a
```

The expected output should look like the screenshot below (target file underlined in red):
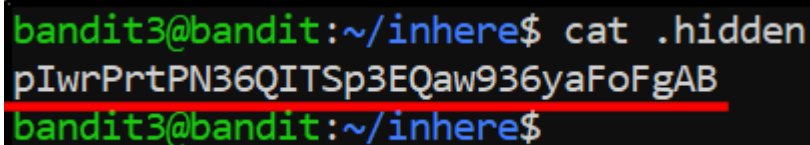

CONTEXT

The **ls** command is able to list out hidden files if it is run with the **-a** argument. In the Linux file system, any file or directory whose name starts with a '**.**' (dot or period) character is considered a "hidden" file or directory and will not be listed out when using the **ls** command unless it is run with the **-a** argument.

> ### Step 7

Use the **cat** command to read the hidden file:

```
cat .hidden
```

The output of this command should look like the screenshot below (flag string underlined in red):

CONTEXT

Although files or directories in Linux can be designated as "hidden" by appending a dot/period to the beginning of the file/directory's name, it is not a secure way to prevent files from being found or accessed.

Step 8

As usual, we will exit the server and record the flag to our password file:

```
exit
echo B4 <FLAG> >> banditPass.txt
```

Part 4 – Learning the File Identification Command With Bandit 4
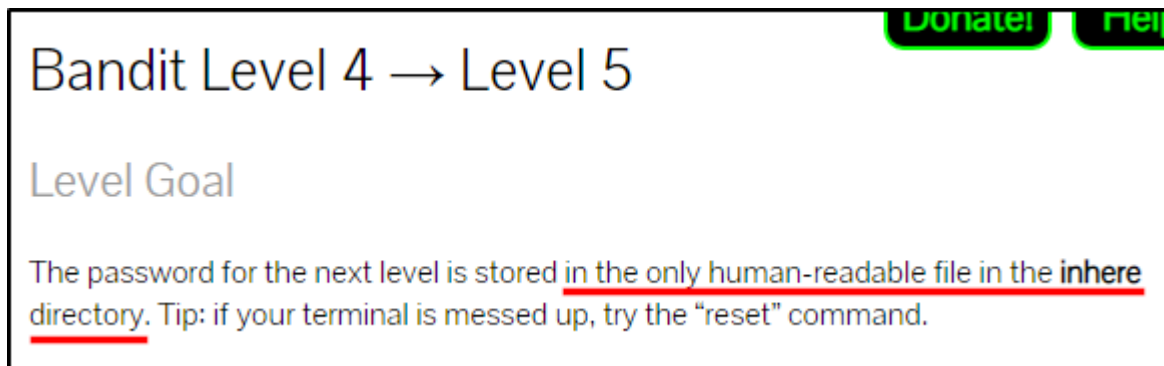
Step 1

Login to the Bandit CTF server as the bandit4 user:

```
ssh bandit4@bandit.labs.overthewire.org -p2220
```
password: obtained in Bandit level 3

Step 2

Check the objectives of Bandit level 4 at the following webpage:

https://overthewire.org/wargames/bandit/bandit5.html

The webpage indicates that there will be files in this level that are not human-readable. This means that they do not contain human-readable "text", but rather "data" that is intended for computer interpretation.

Step 3

We will look for the **inhere** directory with the **ls** command:

```
ls
```

The output of the command should look like the screenshot below (target directory underlined in red):

Step 4

The target directory is here, so let's enter the directory with the **cd** command, then list its contents with the **ls** command:

```
cd inhere; ls
```


Step 5

Let's try reading the first file here with the **cat** command:

```
cat ./-file00
```

The output of the command should look like the screenshot below:

CONTEXT

Nothing to see here. We could check all ten files one by one, but that would be a bit tedious. Instead, let's find a way to speed up the process of looking for the right file in the next step.

Step 6

Use the Linux file command and some special characters the to determine the contents of all of the files in this directory:

```
file ./*
```

The output of this command should be the same as the output below (target file underlined in red):

```
bandit4@bandit:~/inhere$ file ./*
./-file00: data
./-file01: data
./-file02: data
./-file03: data
./-file04: data
./-file05: data
./-file06: data
./-file07: ASCII text
./-file08: data
./-file09: data
bandit4@bandit:~/inhere$
```

CONTEXT

The Linux **file** command will identify the type of content a file contains (text, data, image, archive, etc). It is a very useful tool for doing initial analysis on unknown files.

We should remember that because the file names we're interacting with have special characters in them, so we need to supply a file path (the '**./**' portion of our command).

The special character '*****' (star or asterisk) in our command activates something called filename expansion, which allows our command to apply to all of the files in the directory.

Step 7

Now that we've identified which of the files contains text, let's use the **cat** command to read it:

```
cat ./-file07
```

The output of the command should look like the screenshot below (flag string underlined in red):

```
bandit4@bandit:~/inhere$ cat ./-file07
koReBOKuIDDepwhWk7jZC0RTdopnAYKh
bandit4@bandit:~/inhere$
```

Step 8

Exit the server and copy the flag to our password file:

```
exit
echo B5 <FLAG> >> banditPass.txt
```

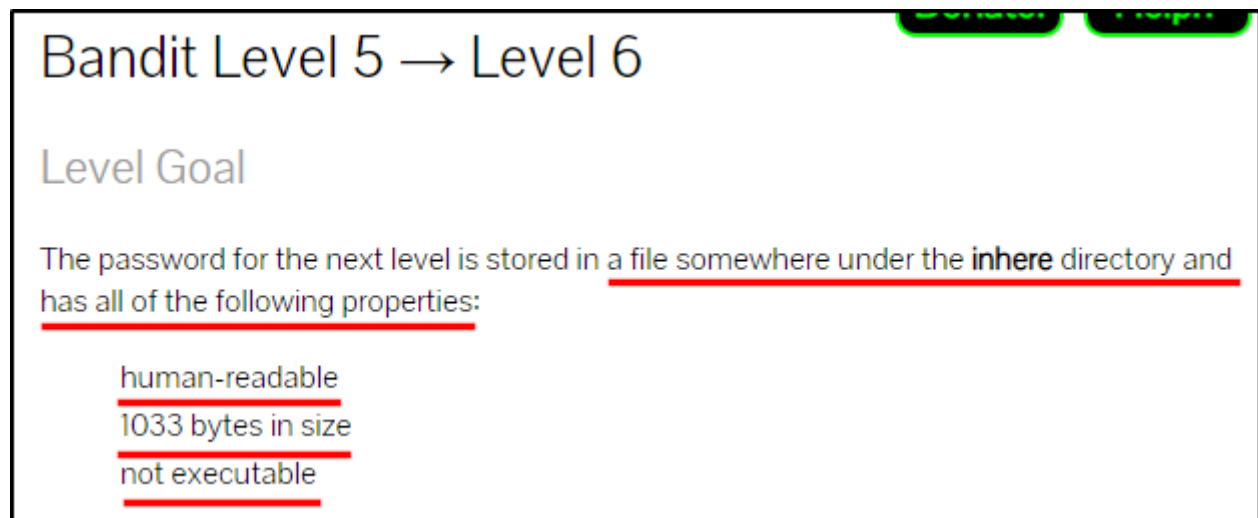# Part 5 – Learning the Find Command With Bandit 5

Step 1

Login to Bandit level 5 with SSH:

```
ssh bandit5@bandit.labs.overthewire.org -p2220
```
password: obtained from Bandit level 4

Step 2

Check the level objectives at the following webpage:

https://overthewire.org/wargames/bandit/bandit6.html

This level of Bandit is about searching for files in the system with specific properties. Let's get started.

Step 3

Let's enter the inhere directory and list out its contents:

```
cd inhere; ls
```

The results of the command should look like the screenshot below:

```
bandit5@bandit:~$ cd inhere; ls
maybehere00  maybehere02  maybehere04  maybehere06  maybehere08  maybehere10  maybehere12  maybehere14  maybehere16  maybehere18
maybehere01  maybehere03  maybehere05  maybehere07  maybehere09  maybehere11  maybehere13  maybehere15  maybehere17  maybehere19
bandit5@bandit:~/inhere$
```

There's twenty directories to search through, so looking through them one by one would be very time consuming, so instead we'll search for the file using the **find** command.

Step 4

Run the **find** command to locate the target file:

```
find ./ -type f -size 1033c ! -executable
```

The output of this command should look like the following screenshot (target file underlined in red):

```
bandit5@bandit:~/inhere$ find ./ -type f -size 1033c ! -executable
./maybehere07/.file2
bandit5@bandit:~/inhere$
```

CONTEXT

The **find** command will match files or directories according to the criteria provided in the arguments to the command. We'll break down the arguments to the command to illustrate exactly what's going on:

```
./                <-- This specifies which directory to look in.
-type f           <-- This specifies to find files only
-size 1033c       <-- This specifies the size of the file
! -executable     <-- This specifies that the file is not executable
```

Step 5

Read the target file:

```
cat ./maybehere07/.file2
```

The output of the command should look like the screenshot below (flag string underlined in red):

```
bandit5@bandit:~/inhere$ cat ./maybehere07/.file2
DXjZPULLxYr17uwoI01bNLQbtFemEgo7
```

Step 6

Exit the server and copy the flag to our password txt file:
```
exit
echo B6 <FLAG> banditPass.txt
```

Summary

While working through the Bandit CTF levels 0 to 5, we learned to use The following commands to work inside of the Linux OS file system:

ls   <-- this command is used to list the contents of a directory
cat <-- this command is used to read the contents of a file
cd   <-- this command is used to change our working directory
file <-- this command is used to identify file types
find <-- this command is used to search for specific files in the system
pwd <-- this command is used to check the current directory

We also learned about cybersecurity CTF games and how to use the SSH program to connect to a server hosting one of these games (Bandit).

Extra Credit

Here are some extra exercises that correspond with the material in today's workshop:

picoCTF – Obedient Cat
https://play.picoctf.org/practice/challenge/147

NOTES: We will need a valid user account to access this website. Use the website's web shell to download the challenge file using **wget** (click on the challenge's second hint).

picoCTF – Magikarp Ground Mission
https://play.picoctf.org/practice/challenge/189

NOTES: We will need a valid user account to access this website. Click on the **Start Instance** button to start the SSH server, then use the website's web shell to login to the SSH server to start the challenge.

Extra Research

Here are some extra articles that correspond with the material in today's workshop:

Differnces Between Windows MacOS and Linux:
https://www.educative.io/answers/differences-between-windows-macos-and-linux-operating-systems

Wikipedia Entry on Command Line Interfaces:
https://en.wikipedia.org/wiki/Command-line_interface

In the next HackerFrogs AfterSchool Linux OS operations workshop, we'll be continuing our Linux CLI journey with the Bandit CTF.

Until next time, HackerFrogs!