• 信息安全技术 •

基于堆栈的 Windows Shellcode 编写方法研究

迟 强， 罗 红， 乔向东

(空军工程大学 电讯工程学院，陕西 西安 710077)

摘 要：为更好地理解与防范缓冲区溢出攻击，对 Windows 平台下 Shellcode 的编写、提取技术及验证方法进行了研究。从概念出发，理清了 Shellcode 与 Exploit 的区别，分析了 Shellcode 的工作原理，介绍了利用 Shellcode 所需的 3 个步骤。在实验的基础上，总结了 Shellcode 的编写方法及提取技术，最后给出了验证 Shellcode 有效性的方法。

关键词：缓冲区溢出；Shellcode；Windows；堆栈；反汇编

中图法分类号：TP309 **文献标识码**：A **文章编号**：1000-7024 (2010) 06-1198-04

Method research of writing Windows Shellcode based on stack

CHI Qiang, LUO Hong, QIAO Xiang-dong

(College of Telecommunication Engineering, Air Force Engineering University, Xi' an 710077, China)

Abstract: To understand and prevent buffer overflow exploit better, the method of writing and validating Shellcode is researched. Firstly, the difference between Shellcode and Exploit is analyzed after introducing the concept. Secondly, the principle of Shellcode and the three steps of exploiting Shellcode are illustrated. Thirdly, based on many experiments, several methods of writing and extracting Shellcode are summarized. Finally, a few measures of validating Shellcode are presented.

Key words: buffer overflow; Shellcode; Windows; stack; disassemble

0 引 言

基于堆栈的缓冲区溢出攻击依然是威胁网络安全的重要手段，溢出发生后，攻击者通常会向目标主机的进程内植入实现特定功能的代码(Shellcode)，并使其执行，对安全系统构成了极大威胁。而缓冲区溢出漏洞的利用，实质上就是使计算机的进程跳入我们编写的 Shellcode 中去执行，分析研究 Shellcode 对于更好的认识与防范缓冲区溢出攻击有重要意义，如文献[1-2]中介绍了通过对 Shellcode 进行检测来防范缓冲区溢出攻击的方法。Shellcode 通常用汇编语言编写并需将其转换成二进制机器码，其内容和长度又经常受到很多苛刻限制，开发和调试难度较大，目前大多数攻击采用的 Shellcode 代码都是来自于 Internet，如通用漏洞测试平台 Metasploit 就提供了一些常用的 Shellcode(如用于绑定端口反向连接、新建用户等)。除了直接使用前人写好的 Shellcode 外，有时我们还需自己动手编写或将其进一步改进成更高质量的符合自己需求的 Shellcode，这就需要对 Shellcode 的编写技术进行研究，本文针对 Windows 平台下 Shellcode 的编写提取方法进行了研究，并给出了 Shellcode 的验证方法。

1 Shellcode 概述

Shellcode 是 1996 年 Aleph One 在其论文《SMASHING THE STACK FOR FUN AND PROFIT》中首次提出的，他把向进程中植入的一段用于获得 shell 的代码称为“Shellcode”。后来，这一概念被继续延用扩充，有了更广义的内涵：泛指在缓冲区溢出攻击中被用于植入进程中的代码。通过精心设计 Shellcode，可以实现许多操作，如弹出消息框、添加用户、上传木马、格式化硬盘等。

与之相关的，还有一个概念——Exploit。研究 Shellcode，有必要弄清两者的关系。Exploit 可被译为“破解”或“利用”，是指利用漏洞将代码植入的过程，包括定位并覆盖函数返回地址，获取进程控制权，把 EIP 传递给 Shellcode 使其执行，而不管 Shellcode 究竟如何实现弹出消息框或是添加新用户等操作。这就像导弹(Exploit)与弹头(Shellcode)的关系，导弹可以实现打击一个特定目标，而导弹上携带的弹头却可以是各种功效的[1]。因此 Exploit 往往是针对特定漏洞的，而 Shellcode 有一定的通用性。

2 Shellcode 的原理

编写 Shellcode 的目的，是为了让目标程序偏离预期方式，按编写者的意图运行。Shellcode 以一组十六进制的字符串数组表示，实质上对应着计算机能够直接执行的机器代码。计算机在执行指令时，都只是执行当前指令指针(EIP)指向的指令。当前指令执行后，EIP 自动加 1，指向下一条指令。若有

收稿日期：2009-04-14；修订日期：2009-06-19。

作者简介：迟强 (1984－)，男，山东莱州人，硕士研究生，研究方向为网络安全与信息对抗； 罗红 (1963－)，男，陕西西安人，博士，副教授，研究方向为网络与信息安全； 乔向东 (1970－)，男，陕西佳县人，博士，副教授，研究方向为信息融合与信息安全。E-mail：cq7216@163.com

JMP、CALL、RET 一类的指令，EIP 会被强行改变成指定的地址，从而完成流程的跳转。编写 Shellcode，是想用 Shellcode 的数据把内存中原有的部分数据覆盖掉，之后设法使计算机的指令指针 EIP 指向 Shellcode 在内存中的开始位置，就可以执行 Shellcode，实现想要的功能[4]。

总结来说，编写一个 Shellcode 并使用之成功 Exploit 需要 3 个步骤：

(1)定位程序溢出点，确定函数返回点的精确位置；

(2)编写一段可实现特殊目的的 Shellcode 代码；

(3)覆盖返回点地址，通过某些特定方式，引导指令指针 EIP 传递给 Shellcode 使其执行。

程序溢出点的定位和返回点的覆盖都属于 Exploit 的过程，这里不作过多研究，本文主要研究 Shellcode 编写的相关技术。

3 Shellcode 编写的关键技术

3.1 Shellcode 的编写方法

Shellcode 是计算机可以直接执行的机器码，编写时要非常严谨，即使是一些细微的差错，也会导致整个 Shellcode 的罢工。下面主要讨论两种 Shellcode 编写的方法：

3.1.1 高级语言反汇编提取法

这种方法的思路是：先用高级语言编写所需的实现特定功能的 Shellcode 程序，然后用编译器(如 VC++6.0)编译、调试并反汇编这个程序(如图 1 所示)，从中整理出汇编代码对应的机器码(opcode)，尽量减小它的体积并使它可注入(生成的代码不包含 NULL 字符)，最后得到 Shellcode。

图 1　程序的反汇编调试

采用这种方法经常会遇到一个问题，由高级语言直接反汇编得到的 Shellcode 往往会含有 NULL(\x00)，由于在字符数组中，NULL 是用来终止字符串的，所以将来把 Shellcode 复制到缓冲区的时候会导致异常。

为解决这个问题，就需要把 NULL 转为 non-null opcode[4]，通常用其它具有相同功能的指令替换那些产生 NULL 的指令(如可用 xor ebx，ebx 来代替 mov ebx，0)；也可直接在 Shellcode 的编写过程中使用不生成 NULL 的指令加上 NULL，由此便产生了下面一种编写 Shellcode 的方法。

3.1.2 低级语言汇编提取法

这种方法的思路是：在用高级语言写完程序之后，不经过编译器的编译与反汇编，直接根据函数调用的过程人工编写相应的汇编代码，最后再提取出机器码。这种方法虽然看似稍显复杂，但可以很好的避免出现 NULL，因此，在实际编写中大都采用这种方法。

Shellcode 的执行过程就是函数调用的过程。与 Linux 操作系统下使用系统中断调用不同，在 Windows 平台下函数的调用首先需要把函数所在的动态链接库 load 进去，然后用堆栈传递参数，最后 CALL 该函数的地址即可。

例如，为实现弹出一个消息框，需要调用这个函数：

MessageBoxA(NULL, "hello,CQ", "hello,CQ", NULL)

编写此函数调用的汇编代码需要 3 个步骤[5]：①装载动态链接库 user32.dll。MessageBoxA 是动态链接库 user32.dll 的导出函数。大多数有图形化操作界面的程序都已经加载了这个库；②在调用前需向栈中按从右至左的顺序压入 MessageBoxA 的 4 个参数；③调用 MessageBoxA 这个函数的入口地址。

为了使示例更简洁，实验时已手动加载了 user32.dll。MessageBoxA 函数的入口地址可以通过 user32.dll 在系统中加载的基址和 MessageBoxA 在库中的偏移相加得到。具体可用 VC++ 6.0 自带的工具“Depends”加载任一有图形界面的程序获得。

如图 2 所示，user32.dll 的基地址为 0x77D10000，MessageBoxA 的偏移地址为 0x000407EA，两者相加就得到了 MessageBoxA 在内存中的入口地址 0x77D507EA。

图 2　计算 MessageBoxA 函数的入口地址

下面就可以根据函数调用的过程人工编写相应的汇编代码了。首先参数入栈，先将 EBX 清零，作为字符串的截断符，然后将“hello,CQ”字符串逆序压入栈中，第 2 和第 3 个参数重复压入指向这个字符串的指针，表示消息框的文本和标题都显示为“hello,CQ”，第 1 个和第 4 个参数都设置为 NULL，最后再 CALL 函数地址。写出汇编代码如下：

```
XOR  EBX,EBX          ;将 EBX 清零作为截断符
PUSH EBX              ;避免“PUSH 0”中的 NULL
PUSH 0x51432C6F
PUSH 0x6C6C6548       ;压入“hello,CQ”字符串
MOV  EAX,ESP          ;在 EAX 中存放字符串指针
PUSH EBX
PUSH EAX
PUSH EAX
PUSH EBX              ;4 个参数从右至左入栈
MOV  EAX,0x77D507EA
CALL EAX              ;调用 MessageBoxA
```

将其十六进制机器码提取出来即可得到 Shellcode：“\x33\xDB\x53\x68\x6F\x2C\x43\x51\x68\x48\x65\x6C\x6C\x8B\xC4\x53\x50\x50\x53\xB8\xEA\x07\xD5\x77\xFF\xD0”。

3.2 Shellcode 的提取方法

在编写 Shellcode 的过程中都用到了提取机器码，其具体实施也有下面 3 种方法：

3.2.1 编译器调试提取法

这种方法是将汇编代码在编译器(如 VC++ 6.0)中用_asm

{}嵌入，编译，按F10调试，调出和汇编代码对应的机器码记录即可。为了记录方便，也可以在汇编内存中直接拷贝，方法是用VC嵌入汇编并按F10进入调试状态后，调出内存窗口，输入eip，内存窗口就会显示从eip开始的数据，就是我们想要的Shellcode代码，如图3所示。

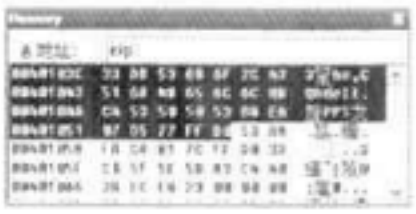

图3 内存中提取机器码

3.2.2 可执行文件提取法

另一种提取Shellcode的方法就是可执行文件提取法，将汇编程序调试成功后，会生成exe可执行文件，将其用Ollydbg来加载，找到相应的机器码即可，如图4所示。

图4 用Ollydbg来提取机器码

3.2.3 C语言直接提取法

除了上面两种方法外，还有一种C语言直接提取法，这个思路最早是由yuange提出，并由hume进一步优化的。它的思路是Shellcode由汇编和C语言混合编写。汇编部分主要完成动态定位函数地址，而C语言部分是完成程序的功能流程。整个程序的本质，就是让编译器为我们生成二进制代码，然后在运行时编码、打印输出[3]。

需要注意的是，上面编写出的Shellcode有一个缺点，就是通用性不好。由于不同版本的操作系统，其动态链接库的加载基址、导出函数的地址都可能不同，使用手工查出的函数地址很可能会在其它计算机上失效，因此需要研究编写可以跨平台使用的通用Shellcode的技术。解决Shellcode的通用性本质上就是要解决函数地址的通用性，因此在实际使用中的Shellcode需要动态地获得自身所需的API函数地址。Win_32平台下Shellcode最常使用的方法，就是通过从进程控制块中找到动态链接库的导出表，并搜索所需的API地址，然后逐一调用[3]。关于通用Shellcode的开发研究，在文献[3]中有详细例解。

4 验证Shellcode的方法

当我们花费半天功夫编写出一个Shellcode，如何确定它们是否正确有效？如何进行测试与调试？这就有必要研究Shellcode的验证方法。

通常有两种方法：第1种实际测试法，就是在实际的溢出程序中，使用提取出来的Shellcode进行测试。第2种Shellcode直接执行法，是把Shellcode数组当成一个函数来执行。其中方法二比较方便易行，下面重点介绍此方法。

这个方法具体实施起来也有几种不同的方式：

4.1 函数指针调用法

打开VC++6.0，新建一个工程和C++源文件，然后把Shellcode拷贝下来作为一个十六进制的字符串数组，最后在main中添加下面一个语句即可。

((void(*)(void)) &Shellcode) ();

这个语句把Shellcode转换成一个参数为空、返回值为空的函数指针，并调用它。执行这一句就相当于执行Shellcode数组里的数据。

我们可以用这段程序运行写好的或搜集到的Shellcode，并调试之。若不能满足需求，也可以在调试的基础上稍作修改，为它增加其它功能。

举例完整代码如下：

```
unsigned char ShellCode[] =
"\xFC\x68\x6A\x0A\x38\x1E\x68\x63\x89\xD1\x4F\x68\x32\
x74\x91\x0C"
"\x8B\xF4\x8D\x7E\xF4\x33\xDB\xB7\x04\x2B\xE3\x66\
xBB\x33\x32\x53"
"\x68\x75\x73\x65\x72\x54\x33\xD2\x64\x8B\x5A\x30\x8B\
x4B\x0C\x8B"
"\x49\x1C\x8B\x09\x8B\x69\x08\xAD\x3D\x6A\x0A\x38\
x1E\x75\x05\x95"
"\xFF\x57\xF8\x95\x60\x8B\x45\x3C\x8B\x4C\x05\x78\x03\
xCD\x8B\x59"
"\x20\x03\xDD\x33\xFF\x47\x8B\x34\xBB\x03\xF5\x99\x0F\
xBE\x06\x3A"
"\xC4\x74\x08\xC1\xCA\x07\x03\xD0\x46\xEB\xF1\x3B\
x54\x24\x1C\x75"
"\xE4\x8B\x59\x24\x03\xDD\x66\x8B\x3C\x7B\x8B\x59\
x1C\x03\xDD\x03"
"\x2C\xBB\x95\x5F\xAB\x57\x61\x3D\x6A\x0A\x38\x1E\
x75\xA9\x33\xDB"
"\x53\x68\x6F\x2C\x43\x51\x68\x48\x65\x6C\x6C\x8B\xC4\
x53\x50\x50"
"\x53\xFF\x57\xFC\x53\xFF\x57\xF8";
int main ()
{
        ((void(*)(void)) &ShellCode) ();
        return 0;
}
```

结果如图5所示。

4.2 返回函数调用法

我们也可以使用这样一段简单的代码对Shellcode进行验证调试：

```
char shellcode[] = "\xFC\x68\x6A\x0A\x38\x1E\x68…..";
                        //欲调试的十六进制机器码
void main ()
{
_asm
{lea eax, shellcode//将 shellcode 的起始地址传递给寄存器
```

图 5　Shellcode 的验证方法 1

```
push eax     //将 eax 中的值入栈
ret}}
```

首先将 Shellcode 的起始地址传递给寄存器 eax，然后将 eax 中的值 push 入栈，最后 ret 指令会将 push 进去的 Shellcode 在栈中的起始地址弹给 EIP，让处理器跳转到栈区去执行 Shellcode，示例如图 6 所示。

图 6　Shellcode 的验证方法 2

4.3　地址直接调用法

根据函数调用原理，也可以直接 call 函数的地址去执行 Shellcode，这里当然是把 Shellcode 当成一个函数来看。具体代码如下：

```
_asm
{
    lea eax, shellcode
    call eax //直接用 call 语句调用 shellcode
}
```

总而言之，方法 2 就是把 Shellcode 数组当成一个函数，设法调用它，使它可以直接执行。

5　结束语

编写 Shellcode 不仅仅要考虑其实现的功能，还要考虑跨平台、健壮性、稳定性、通用性等各方面因素。因此，要编写一段高质量的 Shellcode，还需用到一些高级的编写技术，比如编码技术、"瘦身"技术等，通过对 Shellcode 进行编码，可以有效避免截断、使其符合目标程序对字符的规范要求，还可以突破 IDS 的探测；短小精悍的 Shellcode 可以应用在更小的内存空间中，适应多种多样的缓冲区组织策略，具有更强的通用性。这些技术的研究将是以后工作的重点。

参考文献：

[1] 何乔,吴廖丹,张天刚.基于 shellcode 检测的缓冲区溢出攻击防御技术研究[J].计算机应用,2007,27(5):1045-1049.

[2] 陈悦,薛质,王轶骏.针对 Shellcode 变形规避的 NIDS 检测技术[J].信息安全与通信保密,2007,23(1):99-103.

[3] 王清.0day 安全:软件漏洞分析技术[M].北京:电子工业出版社,2008:72-79.

[4] Arhat.Shellcoder's handbook:Discovering and exploiting security holes [EB/OL]. http://download.csdn.net/down/7046-04/hhincq,2007.

[5] 王炜, 方勇. Q 版缓冲区溢出教程 [EB/OL]. http://down-load.csdn.net/source/802477,2008.

[6] SKAPE. Understanding Windows shellcode [EB/OL]. http://www.hick.org/code/skape/papers/win32-shellcode.pdf,2003.

[7] 谭文,邵坚磊.从汇编语言到 Windows 内核编程[M].北京:电子工业出版社,2008.

[8] 许治坤,王伟,郭添森,等.网络渗透技术[M].北京:电子工业出版社,2005.

(上接第 1197 页)

参考文献：

[1] 中国互联网络信息中心.2008 年 7 月第 21 次中国互联网络发展状况统计报告[Z].

[2] 陈明建.电子邮件协议还原及分析系统的设计与实现[J].计算机应用与研究,2006,20(12):265-270.

[3] 张诚,郝东白.基于正则表达式的 WebMail 监控与审计[J].计算机工程与设计,2008,29(13):3277-3282.

[4] 王佰玲,方滨兴,云晓春.零拷贝报文捕获平台的研究与实现[J].计算机学报,2005,28(1):462-466.

[5] 李旭芳.网络信息审计系统中数据采集的研究与实现[J].计算机工程与设计,2007,28(3):550-552.

[6] 蒋卓明,许榕生.基于内容审计的千兆网络监控系统的设计[J].计算机应用研究,2008,25(4):1114-1116.

[7] Johnson K.Internet E-mail 协议开发指南[M].北京:机械工业出版社,2000:124-128.

[8] 陈明建.高速网络数据的存储及内容监控[D].福州大学,2006.

[9] 李庚.入侵检测中一种新的多模式匹配算法[J].计算机应用研究,2008,25(8):2474-2476.

[10] Yang Dong-Hong,Xu Ke,Cui Yong.An improved Wu-Manber multiple patterns matching algorithm [J]. Journal of Tsinghua University Science and Technology,2006,46(4):5552-5558.

[11] 孙钦东,管晓宏.网络信息内容审计研究的现状及趋势[J].计算机研究与发展,2009,46(8):1241-1250.

基于堆栈的Windows Shellcode编写方法研究

作者： 迟强， 罗红， 乔向东， CHI Qiang， LUO Hong， QIAO Xiang-dong
作者单位： 空军工程大学,电讯工程学院,陕西,西安,710077
刊名： 计算机工程与设计 PKU
英文刊名： COMPUTER ENGINEERING AND DESIGN
年，卷(期)： 2010,31(6)
被引用次数： 1次

参考文献(8条)

1. 何乔;吴廖丹;张天刚 基于shellcode检测的缓冲区溢出攻击防御技术研究[期刊论文]-计算机应用 2007(05)

2. 陈悦;薛质;王轶骏 针对Shellcode变形规避的NIDS检测技术[期刊论文]-信息安全与通信保密 2007(01)

3. 王清 Oday安全:软件漏洞分析技术 2008

4. Arhat Shellcoder's handbook:Discovering and exploiting security holes 2007

5. 王炜;方勇 Q版缓冲区溢出教程 2008

6. SKAPE Understanding Windows shellcode 2003

7. 谭文;邵坚磊 从汇编语言到Windows内核编程 2008

8. 许治坤;王伟;郭添森 周络渗透技术 2005

本文读者也读过(5条)

1. 何乔,吴廖丹,张天刚,HE Qiao,WU Liao-dan,ZHANG Tian-gang 基于shellcode检测的缓冲区溢出攻击防御技术研究[期刊论文]-计算机应用2007,27(5)

2. 戈载,史洪,徐良华,Ge Ji,Shi Hong,Xu Lianghua Shellcode静态检测技术研究[期刊论文]-计算机应用与软件2010,27(2)

3. 和力,吴丽贤,HE Li,WU Li-xian 关于C++虚函数底层实现机制的研究与分析[期刊论文]-计算机工程与设计2008,29(10)

4. 王颖,李祥和,关龙,崔宝江,WANG Ying,LI Xiang-he,GUAN Long,CUI Bao-jiang shellcode攻击与防范技术[期刊论文]-计算机工程2010,36(18)

5. 江亮,王永杰,鲜明,肖顺平,JIANG Liang,WANG Yong-jie,XIAN Ming,XIAO Shun-ping 突破防护措施的shellcode技术研究[期刊论文]-计算机应用研究2007,24(11)

引证文献(1条)

1. 黄惠烽 图书馆局域网缓冲区溢出之shellcode原理分析[期刊论文]-湖北成人教育学院学报 2013(1)