第29卷 第10期 计 算 机 工 程 2003年6月
Vol.29 № 10 Computer Engineering June 2003
· 博士论文 · 文章编号：1000—3428(2003)10 —0005—03 文献标识码：A 中图分类号： TP393.08

缓冲区溢出攻击：原理，防御及检测

蒋卫华，李伟华，杜 君
(西北工业大学计算机科学与工程系，西安710072)

摘 要：给出了缓冲区溢出的原理，分析了利用缓冲区溢出漏洞进行网络攻击的方法及其特征；从程序编写、程序检测、数据结构设计以及程序执行控制等多个角度对防止缓冲区溢出攻击进行了分析，提出了遏制利用缓冲区溢出漏洞进行攻击的一些方法。

关键词：缓冲区溢出；堆栈溢出；黑客攻击；防御；缓冲区检测；缓冲区不可执行

Buffer Overflow Attack: Theory, Recovery and Detection

JIANG Weihua，LI Weihua，DU Jun
(Department of Computer Science & Engineering, Northwestern Polytechnical University, Xi'an 710072)

【**Abstract**】This paper first describes the technical principles and features of network attack based on buffer overflow, then analyzes the recovery methods of buffer overflow from programming, detecting, data structure designing and program executing. At last, it presents some methods to prevent this kind of attacks.

【**Key words**】Buffer overflow; Stack overflow; Hacker attack; Recovery; Buffer check; Buffer non-execution

网络防范技术的日益成熟，使木马、病毒这类恶意代码的植入变得困难。网络黑客开始针对系统和程序自身存在的漏洞，编写相应的攻击程序。其中最常见的就是对缓冲区溢出漏洞的攻击，而在诸多缓冲区溢出中又以堆栈溢出的问题最有代表性。目前，利用缓冲区溢出漏洞进行的攻击已经占到了整个网络攻击次数的一半以上。

世界上第一个缓冲区溢出攻击——Morris蠕虫，发生在十几年前，曾造成了全球6000多台网络服务器瘫痪。事实上，缓冲区溢出漏洞被攻击的现象目前已越来越普遍，各种操作系统上出现的此种漏洞都数不胜数。例如，在BSD上存在打印守护进程远程缓冲区溢出漏洞；在Sun OS上的Solaris whodo本地缓冲区溢出漏洞；世界上第一个Linux病毒Reman，其实就是一个缓冲区溢出攻击程序；而Windows下IIS4、IIS5某些版本在处理超长文件名时，存在缓冲区溢出漏洞。

对缓冲区溢出漏洞攻击，可以导致程序运行失败、系统崩溃以及重新启动等后果。更为严重的是，可以利用缓冲区溢出执行非授权指令，甚至取得系统特权，进而进行各种非法操作。如何防止和检测出利用缓冲区溢出漏洞进行的攻击，就成为防御网络入侵以及入侵检测的重点之一。

1 缓冲区溢出的分析

1.1 缓冲区溢出的原理

简单地说，缓冲区溢出的原因是由于字符串处理函数(gets,strcpy等)没有对数组的越界加以监视和限制，结果覆盖了老的堆栈数据。

在计算机内的程序是按以下形式存储的，见图1。

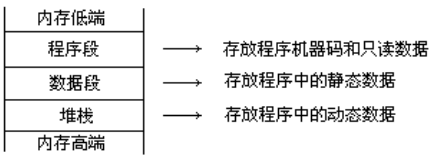

图 1 程序在内存中的存储

从图1可以看出，输入的形参等数据存放在堆栈中，程序是从内存低端向内存高端按顺序执行的，由于堆栈的生长方向与内存的生长方向相反，因此在堆栈中压入的数据超过预先给堆栈分配的容量时，就会出现堆栈溢出，从而使得程序运行失败；如果发生栈溢出的是大型程序还有可能会导致系统崩溃。

我们来看一段简单程序的执行过程中对堆栈的操作和溢出的产生过程。

```
#include <stdio.h>
int main()
char name[16];
gets(name);
for(int i=0;i<16&&name[i];i++)
printf(,name[i]);  }
```

编译上述代码，输入“hello world！”结果会输出hello world！,其中对堆栈的操作是先在栈底压入返回地址，接着将栈指针EBP入栈，此时EBP等于现在的ESP，之后ESP减16，即向上增长16个字节，用来存放name[]数组，现在堆栈的布局如图2。

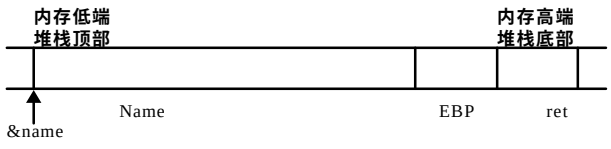

图 2 程序运行之初堆栈的状态

执行完gets(name)之后，堆栈中的内容如见图3。

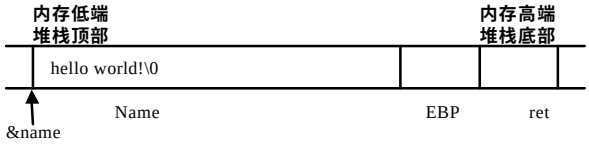

图 3 运行完gets(name)后堆栈的状态

基金项目：国家高技术研究发展计划项目(2001AA142100)；教育部博士点基金项目；航空科学基金项目
作者简介：蒋卫华(1973－)，男，讲师、博士，主研方向：信息与网络安全；李伟华，教授、博导；杜 君，硕士
收稿日期：2002-07-05

最后，从main返回，弹出ret里的返回地址并赋值给EIP，CPU继续执行EIP所指向的命令。

如果我们输入的字符串长度超过16个字节，例如输入：hello world!AAAAAAAAAAA，则当执行完gets(name)之后，堆栈的情况如图4。

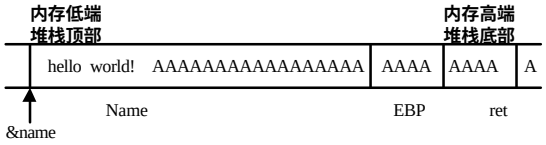

图4 缓冲区溢出状态

由于输入的字符串太长，name数组容纳不下，只好向堆栈的底部方向继续写‘A’。这些‘A’覆盖了堆栈的老的元素，从图4可以看出，EBP，ret都已经被‘A’覆盖了。从main返回时，就必然会把‘AAAA’的ASCII码——0x41414141视作返回地址，CPU会试图执行0x41414141处的指令，结果出现难以预料的后果，这样就产生了一次堆栈溢出。假如使用的操作系统为Win9X的话，会得到那个经典的“该程序执行了非法操作”的对话框。

1.2 溢出字符串的特征

在分析溢出字符串的特征之前，需要先大致了解溢出字符串的编写。溢出字符串由若干个普通的ASCII码字符组成，在攻击者确定了缓冲区大小和缓冲区相对于堆栈开始地址的便宜量时，溢出字符串只有一个越界特征。溢出字符串的形式如图5所示。

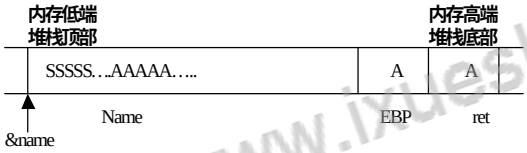

图5 溢出字符串的形式

这时它和普通的字符串几乎没有什么差别，是最难判断的情况(几乎没有好办法对付这种情况)。但当攻击者不清楚缓冲区相对堆栈开始地址的偏移量，为了提高跳转地址的命中率(使得ret的值等于shellcode的入口地址)，一般攻击者会在其溢出字符串和shellcode前加入若干个NOP，它的作用就是什么也不做，仅跳过一个CPU周期。用来溢出的字符串会变成图6所示的结构。

图6 增加若干NOP指令后的溢出字符串

而在溢出字符串最前面的若干NOP指令将成为识别这种攻击的一大特征。

1.3 缓冲区溢出漏洞的产生原因

缓冲区溢出的根本原因在于C语言本身的一些特性。从数据结构的角度来说，最根本的原因是由于char * (或char[])数据结构的存在，导致了一系列字符串存储以及操作上的问题。而直接的原因则是“由于字符串处理函数(gets,strcpy等等)没有对数组的越界加以监视和限制”。C中大多数缓冲区溢出问题可以追溯到标准函数库，直接的原因是不进行自变量检查和使用一些有问题的字符串操作函数(strcpy、strcat、sprinf和gets)。程序编写者的经验不足和粗心大意使得缓冲区溢出漏洞几乎无处不在，导致程序健壮性不够，为缓冲区溢出攻击留下了隐患。特别是由于Internet的迅速发展，各种网络应用程序层出不穷，而其中一个个缓冲区溢出漏洞则给整个系统带来了极大的安全隐患，为黑客攻击打开方便之门。

1.4 缓冲区溢出漏洞的危害性

缓冲区溢出漏洞比其他一些黑客攻击手段更具有破坏力和隐蔽性。这也是利用缓冲区溢出漏洞进行攻击日益普遍的原因。它极容易使服务程序停止运行，服务器死机甚至删除服务器上的数据。它的隐蔽性主要表现在下面几点：首先，漏洞被发现之前一般程序员是不会意识到自己的程序存在漏洞(漏洞的发现者往往并非编写程序的程序员)，从而疏忽监测；其次，shellcode都很短，执行时间也非常短，很难在执行过程中被发现；第三，由于漏洞存在于防火墙内部，攻击者所发送的字符串一般情况下防火墙不会阻拦，而攻击者通过执行shellcode所获得的是本来不被允许或没有权限的操作，在防火墙看来也是合理合法的。防火墙在对远程缓冲区溢出攻击的监测方面有先天的不足；第四，一个完整的shellcode的执行并不一定会使系统报告错误，并可能不影响正常程序的运行；第五，攻击的随机性和不可预测性使得防御攻击变得异常艰难，而没有攻击时，攻击程序并不会有什么变化(这和木马有着本质的区别)，这也是堆栈溢出最难被发现的原因；最后，缓冲区溢出漏洞的普遍存在，使得针对这种漏洞的攻击防不胜防(各种补丁程序也可能存在着这种漏洞)。

另外，还存在着攻击者故意散布存在漏洞的应用程序的可能。攻击者还可以借用木马植入的方法，故意在被攻击者的系统中留下存在漏洞的程序，这样做不会因为含有非法字段而被防火墙拒绝；或者利用病毒传播的方式来传播有漏洞的程序，和病毒不同的是，它在一个系统中只留下一份拷贝(要发现这种情况几乎是不可能的)。

2 防范及检测方法

各种原因产生了大量存在漏洞的程序，而且利用缓冲区溢出攻击主机也时有发生。现在，怎样防范这种危害巨大的攻击手段，已成为网络安全方面一个很重要的研究内容。

2.1 编写程序中应该时刻注意的问题

程序员有责任和义务养成安全编程的思想，应该熟悉那些可能会产生漏洞或需慎用的函数，清楚那些在编程中要小心使用的函数(特别是在使用C语言时)，例如：gets()、strcpy()等等。

在软件测试阶段，要专门对程序中的每个缓冲区作边界检查和溢出检测。但是，由于程序编写者的经验不足和测试工作不够全面、充分，目前还不可能完全避免缓冲区溢出漏洞，因此这些漏洞在已经使用以及正在开发的软件中还是有存在的可能，还需要在使用软件时，对它做实时的监测。

2.2 使用安全语言编写程序

应使用Java等安全的语言编写程序，因为Java在对缓冲区进行操作时，有相应的边界检查，所以可以有效地防止缓冲区溢出漏洞的产生。但是，Java也并非绝对安全，Java的解释器是用C语言编写的，而C并不是一种安全的语言，所以Java解释器还是可能存在缓冲区溢出漏洞并受到攻击。

2.3 改进编译器

改进编译器的主要思想是在编译器中增加边界检查以及保护堆栈的功能，使得含有漏洞的程序和代码段无法通过编译。针对gcc编译器的很多补丁就提供了这些功能，比如说Stackguard。

2.4 利用人工智能的方法检查输入字段

黑客利用缓冲区溢出漏洞进行攻击时，必须将其设计的溢出字符串包含在输入字符串中。如果能检测到输入字段中存在非法字段，就可以将黑客的攻击记录下来，以便防范。上面所介绍的溢出字符串的设计，就可以利用其特征来建立规则集， 使用模式匹配 、 人工智能的方法来监测缓冲区(图7)。例如，可以检查字符流中是否存在大量的NOP字段(特别是在字符串的头部)，来确定是否有缓冲区溢出攻击存在。又如：对于输入串只需要可显示字符时(例如留言簿、URL等)，可以将不可显示字符归入非法字符集；还可以针对字符的排列顺序而规定相应的规则集等等。

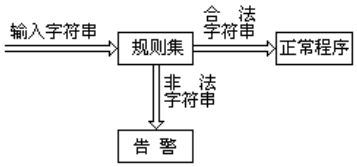

图 7 利用人工智能的方法进行字符串检查

但是这种方法不可能完全消除缓冲区溢出漏洞(人工智能的局限性)。而且规则集怎么设置才会使误报率和漏报率更低，现在还没有一个良好的解决方案。

2.5 对堆栈栈底进行实时的监测

监测一个堆栈应从其被建立到其消亡的全过程，需要监测的内容有堆栈的标志、栈底的地址、栈底存放的内容、被压入栈的返回地址和EBP的值、可执行的压栈操作次数——栈的大小等等。这些内容可以从操作系统获得，并需要监测CPU的状态。可以用两种方法实现对堆栈的实时监测。

(1)缓冲区溢出漏洞攻击的目的是修改栈底的返回地址。据此，可以先建立一张表对栈底内容实行监控，模型如表1所示。

表 1 堆栈监控表

堆栈标志		…
建立堆栈时栈底内容		…
弹出堆栈时栈底内容		…
溢出标志位		…

只要比较堆栈建立时的栈底地址和栈操作结束时的栈底地址，如果相同则置“溢出标志位”表项为“0”，正常返回，不做干预；如果不同，那么可以肯定堆栈发生了溢出，则置相应表项——“出标志位”为“1”，报告错误，并填写相应的日志，禁止按栈内容返回，等待管理员处理。

(2)可以对输入字符串的长度进行监控。同样可以建立一张表，模型如表2所示。

表 2 字符串长度监控表

堆栈标志		…
可执行的压栈操作数		…
已执行的压栈操作数		…
溢出标志位		…

从表中可以很清楚地知道输入字符串的长度是否超过了缓冲区的长度，并通过溢出标志位的值来对堆栈的使用情况进行实时监控。

对堆栈栈底进行实时监测可以有效地防止缓冲区溢出攻击，但其自身也有缺点，即需要大量的系统资源，并会降低程序执行的效率。这种方法可以作为一种对软件进行测试的工具。此外，由于这种方法是较低层的，程序可移植性不强。而且如果程序编写不得当，甚至可能会和操作系统发生冲突，从而导致各种问题的出现。

2.6 堆栈不可执行

这种方法已经在很多种操作系统上有了相应的补丁，但它也不是一个万全之策，既然不可能在堆栈段执行程序，那么就将溢出字符串写入到数据段区或程序段区，这样就仍然可以执行。

2.7 修改现在缓冲区的数据结构

以上这些方法各有各的优点和缺点，但是仅用其中的一种方法或几种方法，并不能够完全杜绝缓冲区溢出漏洞。在防御缓冲区溢出攻击时，应综合使用其中的几种方法，才可以达到良好的效果。然而，这些方法都是一些治标不治本的方法，要从根本上解决缓冲区溢出漏洞的问题，必须从数据结构的角度来考虑问题。

上面我们曾提到，其实绝大多数的缓冲区溢出漏洞，其根本原因就是C语言中的char * 数据结构。由于这一数据结构以及与之相关的各种函数的广泛应用，导致各种应用程序中的缓冲区溢出漏洞层出不穷。而大量业已存在并正在运行的C程序代码，又使得完全消除这一漏洞几乎不可能。

事实上，要从根本上解决缓冲区溢出漏洞，必须从修改缓冲区的数据结构入手。只要有了安全的数据结构，就能构建出安全的函数和程序，从而防止由于数据结构上的不合理而造成的安全隐患。在C++中提倡使用的String函数库，正是针对C语言中的这一弱点而开发的。

3 结束语

缓冲区溢出是一种系统攻击的手段，通过往程序的缓冲区写入超出其长度的内容，造成缓冲区的溢出，从而破坏程序的堆栈，使程序转而执行非预期指令，以达到攻击的目的。目前，在Internet上利用缓冲区溢出进行攻击的行为已经相当普遍。本文从缓冲区溢出的原理入手，指出缓冲溢出漏洞的产生原因，说明其危害性，并分析了现有的防范缓冲区溢出的措施，提出了对防范缓冲区溢出的一些方法。

参考文献

1 Wagle C C,Pu C,Beattie S,et al.Buffer Overflows:Attacks and Defenses for the Vulnerability of the Decade. DARPA Information Survivability Conference and Exposition,2000-01

2 McGraw G,Viega J.Make Your Software Behave:Learning the Basics of Buffer Overflows:Get Reacquainted with the Single Biggest Threat to Software Security.Reliable Software Technologies,2000-03-01

3 Aleph One:Smashing the Stack for Fun and Profit.URL:http://www.shmoo.com/phrack/Phrack49/p49-14,1996-11-08

阅读此文的还阅读了：

26. Sybil攻击原理和防御措施

27. DDoS攻击原理及检测防御技术

28. 浅谈拒绝服务攻击的原理与防御

29. 浅析ARP攻击原理及其防御

30. 从检测到防御

31. CC攻击的原理与防御

32. 缓冲区溢出攻击的原理与防范

33. 浅析CC攻击的原理和防御

34. 缓冲区溢出攻击代码检测与防御技术研究

35. APT检测及防御

36. 网络空间拟态防御原理简介(上)

37. DDoS攻击原理及防御措施

38. 缓冲区溢出攻击原理和现有检测技术

39. APT攻击检测与防御

40. DDoS攻击原理及防御方法研究

41. DDoS攻击原理及防御

42. 浅谈ARP攻击原理及防御对策

43. APT攻击检测与防御

44. DDoS攻击原理和防御方法研究

45. DDoS攻击的原理及防御对策

46. 僵尸网络原理及其防御技术

47. 缓冲区溢出攻击及防御

48. DDoS检测防御研究

49. DDoS攻击原理及防御技术

50. APT攻击检测与防御