缓冲区溢出攻击

缓冲区溢出攻击是利用缓冲区溢出漏洞所进行的攻击行动。缓冲区溢出是一种非常普遍、非常危险的漏洞，在各种操作系统、应用软件中广泛存在。利用缓冲区溢出攻击，可以导致程序运行失败、系统关机、重新启动等后果。更为严重的是，可以利用它执行非授权指令，甚至可以取得系统特权，进而进行各种非法操作。缓冲区溢出攻击有多种英文名称：buffer overflow，buffer overrun，smash the stack，trash the stack，scribble the stack，mangle the stack，memory leak，overrun screw；它们指的都是同一种攻击手段。第一个缓冲区溢出攻击--Morris 蠕虫，发生在二十年前，它曾造成了全世界 6000 多台网络服务器瘫痪。

1.概念

缓冲区溢出是指当计算机向缓冲区内填充数据位数时超过了缓冲区本身的容量，溢出的数据覆盖在合法数据上，理想的情况是：程序检查数据长度并不允许输入超过缓冲区长度的字符，但是绝大多数程序都会假设数据长度总是与所分配的储存空间想匹配，这就为缓冲区溢出埋下隐患。操作系统所使用的缓冲区，又被称为“堆栈”，在各个操作进程之间，指令会被临时储存在“堆栈”当中，“堆栈”也会出现缓冲区溢出。

2.危害

在当前网络与分布式系统安全中，被广泛利用的 50%以上都是缓冲区溢出，其中最著名的例子是 1988 年利用 fingerd 漏洞的蠕虫。而缓冲区溢出中，最为危险的是堆栈溢出，因为入侵者可以利用堆栈溢出，在函数返回时改变返回程序

的地址，让其跳转到任意地址，带来的危害一种是程序崩溃导致拒绝服务，另外一种就是跳转并且执行一段恶意代码，比如得到 shell，然后为所欲为。

3.缓冲区攻击

一. 缓冲区溢出的原理

通过往程序的缓冲区写超出其长度的内容，造成缓冲区的溢出，从而破坏程序的堆栈，使程序转而执行其它指令，以达到攻击的目的。造成缓冲区溢出的原因是程序中没有仔细检查用户输入的参数。例如下面程序：

```
void function(char *str) {
char buffer[16];
strcpy(buffer,str);
}
```

上面的 strcpy()将直接吧 str 中的内容 copy 到 buffer 中。这样只要 str 的长度大于 16，就会造成 buffer 的溢出，使程序运行出错。存在象 strcpy 这样的问题的标准函数还有 strcat()，sprintf()，vsprintf()，gets()，scanf()等。

当然，随便往缓冲区中填东西造成它溢出一般只会出现“分段错误”（Segmentation fault），而不能达到攻击的目的。最常见的手段是通过制造缓冲区溢出使程序运行一个用户 shell，再通过 shell 执行其它命令。如果该程序属于 root 且有 suid 权限的话，攻击者就获得了一个有 root 权限的 shell，可以对系统进行任意操作了。

缓冲区溢出攻击之所以成为一种常见安全攻击手段其原因在于缓冲区溢出漏洞太普遍了，并且易于实现。而且，缓冲区溢出成为远程攻击的主要手段其原因在于缓冲区溢出漏洞给予了攻击者他所想要的一切：植入并且执行攻击代码。

被植入的攻击代码以一定的权限运行有缓冲区溢出漏洞的程序，从而得到被攻击主机的控制权。

在 1998 年 Lincoln 实验室用来评估入侵检测的的 5 种远程攻击中，有 2 种是缓冲区溢出。而在 1998 年 CERT 的 13 份建议中，有 9 份是是与缓冲区溢出有关的，在 1999 年，至少有半数的建议是和缓冲区溢出有关的。在 Bugtraq 的调查中，有 2/3 的被调查者认为缓冲区溢出漏洞是一个很严重的安全问题。

缓冲区溢出漏洞和攻击有很多种形式，会在第二节对他们进行描述和分类。相应地防卫手段也随者攻击方法的不同而不同，将在第四节描述，它的内容包括针对每种攻击类型的有效的防卫手段。

二、缓冲区溢出的漏洞和攻击

缓冲区溢出攻击的目的在于扰乱具有某些特权运行的程序的功能，这样可以使得攻击者取得程序的控制权，如果该程序具有足够的权限，那么整个主机就被控制了。一般而言，攻击者攻击 root 程序，然后执行类似“exec(sh)”的执行代码来获得 root 权限的 shell。为了达到这个目的，攻击者必须达到如下的两个目标：

1. 在程序的地址空间里安排适当的代码。

2. 通过适当的初始化寄存器和内存，让程序跳转到入侵者安排的地址空间执行。

根据这两个目标来对缓冲区溢出攻击进行分类。在二.1 节，将描述攻击代码是如何放入被攻击程序的地址空间的。在二.2 节，将介绍攻击者如何使一个程序的缓冲区溢出，并且执行转移到攻击代码（这个就是“溢出”的由来）。在二.3 节，将综合前两节所讨论的代码安排和控制程序执行流程的技术。

二.1 在程序的地址空间里安排适当的代码的方法

有两种在被攻击程序地址空间里安排攻击代码的方法：

1、植入法：

攻击者向被攻击的程序输入一个字符串，程序会把这个字符串放到缓冲区里。这个字符串包含的资料是可以在这个被攻击的硬件平台上运行的指令序列。在这里，攻击者用被攻击程序的缓冲区来存放攻击代码。缓冲区可以设在任何地方：堆栈（stack，自动变量）、堆（heap，动态分配的内存区）和静态资料区。

2、利用已经存在的代码：

有时，攻击者想要的代码已经在被攻击的程序中了，攻击者所要做的只是对代码传递一些参数。比如，攻击代码要求执行“exec（“/bin/sh”）”，而在 libc 库中的代码执行“exec (arg)”，其中 arg 使一个指向一个字符串的指针参数，那么攻击者只要把传入的参数指针改向指向”/bin/sh”。

二.2 控制程序转移到攻击代码的方法

所有的这些方法都是在寻求改变程序的执行流程，使之跳转到攻击代码。最基本的就是溢出一个没有边界检查或者其它弱点的缓冲区，这样就扰乱了程序的正常的执行顺序。通过溢出一个缓冲区，攻击者可以用暴力的方法改写相邻的程序空间而直接跳过了系统的检查。

分类的基准是攻击者所寻求的缓冲区溢出的程序空间类型。原则上是可以任意的空间。实际上，许多的缓冲区溢出是用暴力的方法来寻求改变程序指针的。这类程序的不同之处就是程序空间的突破和内存空间的定位不同。主要有以下三种：

1、活动纪录（Activation Records）：

每当一个函数调用发生时，调用者会在堆栈中留下一个活动纪录，它包含了函数结束时返回的地址。攻击者通过溢出堆栈中的自动变量，使返回地址指向攻击代码。通过改变程序的返回地址，当函数调用结束时，程序就跳转到攻击者设定的地址，而不是原先的地址。这类的缓冲区溢出被称为堆栈溢出攻击（Stack Smashing Attack），是目前最常用的缓冲区溢出攻击方式。

2、函数指针（Function Pointers）：

函数指针可以用来定位任何地址空间。例如：“void (* foo)()”声明了一个返回值为 void 的函数指针变量 foo。所以攻击者只需在任何空间内的函数指针附近找到一个能够溢出的缓冲区，然后溢出这个缓冲区来改变函数指针。在某一时刻，当程序通过函数指针调用函数时，程序的流程就按攻击者的意图实现了。它的一个攻击范例就是在 Linux 系统下的 superprobe 程序。

3、长跳转缓冲区（Longjmp buffers）：

在 C 语言中包含了一个简单的检验/恢复系统 称为 setjmp/longjmp。意思是在检验点设定“setjmp(buffer)”，用“longjmp(buffer)”来恢复检验点。然而，如果攻击者能够进入缓冲区的空间，那么“longjmp(buffer)”实际上是跳转到攻击者的代码。象函数指针一样，longjmp 缓冲区能够指向任何地方，所以攻击者所要做的就是找到一个可供溢出的缓冲区。一个典型的例子就是 Perl 5.003 的缓冲区溢出漏洞；攻击者首先进入用来恢复缓冲区溢出的的 longjmp 缓冲区，然后诱导进入恢复模式，这样就使 Perl 的解释器跳转到攻击代码上了。

二.3 代码植入和流程控制技术的综合分析

最简单和常见的缓冲区溢出攻击类型就是在一个字符串里综合了代码植入和活动纪录技术。攻击者定位一个可供溢出的自动变量，然后向程序传递一个很大的字符串，在引发缓冲区溢出，改变活动纪录的同时植入了代码。这个是由 Levy 指出的攻击的模板。因为 C 在习惯上只为用户和参数开辟很小的缓冲区，因此这种漏洞攻击的实例十分常见。

代码植入和缓冲区溢出不一定要在在一次动作内完成。攻击者可以在一个缓冲区内放置代码，这是不能溢出的缓冲区。然后，攻击者通过溢出另外一个缓冲区来转移程序的指针。这种方法一般用来解决可供溢出的缓冲区不够大（不能放下全部的代码）的情况。

如果攻击者试图使用已经常驻的代码而不是从外部植入代码，他们通常必须把代码作为参数调用。举例来说，在 libc（几乎所有的 C 程序都要它来连接）中的部分代码段会执行“exec(something)”，其中 somthing 就是参数。攻击者然后使用缓冲区溢出改变程序的参数，然后利用另一个缓冲区溢出使程序指针指向 libc 中的特定的代码段。

三、 缓冲区溢出攻击的实验分析

2000 年 1 月，Cerberus 安全小组发布了微软的 IIS 4/5 存在的一个缓冲区溢出漏洞。攻击该漏洞，可以使 Web 服务器崩溃，甚至获取超级权限执

行任意的代码。目前，微软的 IIS 4/5 是一种主流的 Web 服务器程序；因而，该缓冲区溢出漏洞对于网站的安全构成了极大的威胁；它的描述如下：

浏览器向 IIS 提出一个 HTTP 请求，在域名（或 IP 地址）后，加上一个文件名，该文件名以“.htr”做后缀。于是 IIS 认为客户端正在请求一个“.htr”文件，“.htr”扩展文件被映像成 ISAPI（Internet Service API）应用程序，IIS 会复位向所有针对“.htr”资源的请求到 ISM.DLL 程序 ，ISM.DLL 打开这个文件并执行之。

浏览器提交的请求中包含的文件名存储在局部变量缓冲区中，若它很长，超过 600 个字符时，会导致局部变量缓冲区溢出，覆盖返回地址空间，使 IIS 崩溃。更进一步，在如图 1 所示的 2K 缓冲区中植入一段精心设计的代码，可以使之以系统超级权限运行。

四、缓冲区溢出攻击的防范方法

缓冲区溢出攻击占了远程网络攻击的绝大多数，这种攻击可以使得一个匿名的 Internet 用户有机会获得一台主机的部分或全部的控制权。如果能有效地消除缓冲区溢出的漏洞，则很大一部分的安全威胁可以得到缓解。

目前有四种基本的方法保护缓冲区免受缓冲区溢出的攻击和影响。在四.1 中介绍了通过操作系统使得缓冲区不可执行，从而阻止攻击者植入攻击代码。在四.2 中介绍了强制写正确的代码的方法。在四.3 中介绍了利用编译器的边界检查来实现缓冲区的保护。这个方法使得缓冲区溢出不可能出现，从而完全消除

了缓冲区溢出的威胁 ,但是相对而言代价比较大。在四.4 中介绍一种间接的方法，这个方法在程序指针失效前进行完整性检查。虽然这种方法不能使得所有的缓冲区溢出失效，但它能阻止绝大多数的缓冲区溢出攻击。然后在四.5，分析这种保护方法的兼容性和性能优势。

四.1 非执行的缓冲区

通过使被攻击程序的数据段地址空间不可执行 ,从而使得攻击者不可能执行被植入被攻击程序输入缓冲区的代码 ,这种技术被称为非执行的缓冲区技术。在早期的 Unix 系统设计中，只允许程序代码在代码段中执行。但是近来的 Unix 和 MS Windows 系统由于要实现更好的性能和功能，往往在数据段中动态地放入可执行的代码，这也是缓冲区溢出的根源。为了保持程序的兼容性，不可能使得所有程序的数据段不可执行。

但是可以设定堆栈数据段不可执行，这样就可以保证程序的兼容性。Linux 和 Solaris 都发布了有关这方面的内核补丁。因为几乎没有任何合法的程序会在堆栈中存放代码，这种做法几乎不产生任何兼容性问题，除了在 Linux 中的两个特例，这时可执行的代码必须被放入堆栈中：

（1）信号传递：

Linux 通过向进程堆栈释放代码然后引发中断来执行在堆栈中的代码来实现向进程发送 Unix 信号。非执行缓冲区的补丁在发送信号的时候是允许缓冲区可执行的。

（2）GCC 的在线重用：

研究发现 gcc 在堆栈区里放置了可执行的代码作为在线重用之用。然而，关闭这个功能并不产生任何问题，只有部分功能似乎不能使用。

非执行堆栈的保护可以有效地对付把代码植入自动变量的缓冲区溢出攻击，而对于其它形式的攻击则没有效果。通过引用一个驻留的程序的指针，就可以跳过这种保护措施。其它的攻击可以采用把代码植入堆或者静态数据段中来跳过保护。

四.2 编写正确的代码

编写正确的代码是一件非常有意义的工作 ,特别象编写 C 语言那种风格自由而容易出错的程序 ,这种风格是由于追求性能而忽视正确性的传统引起的。尽管花了很长的时间使得人们知道了如何编写安全的程序 ,具有安全漏洞的程序依旧出现。因此人们开发了一些工具和技术来帮助经验不足的程序员编写安全正确的程序。

最简单的方法就是用 grep 来搜索源代码中容易产生漏洞的库的调用，比如对 strcpy 和 sprintf 的调用，这两个函数都没有检查输入参数的长度。事实上，各个版本 C 的标准库均有这样的问题存在。

此外，人们还开发了一些高级的查错工具，如 fault injection 等。这些工具的目的在于通过人为随机地产生一些缓冲区溢出来寻找代码的安全漏洞。还有一些静态分析工具用于侦测缓冲区溢出的存在。

虽然这些工具帮助程序员开发更安全的程序，但是由于 C 语言的特点，这些工具不可能找出所有的缓冲区溢出漏洞。所以，侦错技术只能用来减少缓冲区溢出的可能，并不能完全地消除它的存在。

如果把一加仑的水注入容量为一品脱的容量中，水会四处冒出，这时你就会充分理解溢出的含义。同样的道理，在计算机内部，如果你向一个容量有限的内存空间里存储过量数据，这时数据也会溢出存储空间。输入数据通常被存放在一个临时空间内，这个临时存放空间被称为缓冲区，缓冲区的长度事先已经被程序或者*作系统定义好了。

何为缓冲区溢出

缓冲区溢出是指当计算机程序向缓冲区内填充的数据位数超过了缓冲区本身的容量。溢出的数据覆盖在合法数据上。理想情况是，程序检查数据长度并且不允许输入超过缓冲区长度的字符串。但是绝大多数程序都会假设数据长度总是与所分配的存储空间相匹配，这就为缓冲区溢出埋下隐患。*作系统所使用的缓冲区又被称为堆栈，在各个*作进程之间，指令被临时存储在堆栈当中，堆栈也会出现缓冲区溢出。

当一个超长的数据进入到缓冲区时，超出部分就会被写入其他缓冲区，其他缓冲区存放的可能是数据、下一条指令的指针，或者是其他程序的输出内容，

这些内容都被覆盖或者破坏掉。可见一小部分数据或者一套指令的溢出就可能导致一个程序或者*作系统崩溃。

溢出根源在于编程

缓冲区溢出是由编程错误引起的。如果缓冲区被写满，而程序没有去检查缓冲区边界，也没有停止接收数据，这时缓冲区溢出就会发生。缓冲区边界检查被认为是不会有收益的管理支出，计算机资源不够或者内存不足是编程者不编写缓冲区边界检查语句的理由，然而摩尔定律已经使这一理由失去了存在的基础，但是多数用户仍然在主要应用中运行十年甚至二十年前的程序代码。

缓冲区溢出之所以泛滥，是由于开放源代码程序的本质决定的。一些编程语言对于缓冲区溢出是具有免疫力的，例如 Perl 能够自动调节字节排列的大小，Ada95 能够检查和阻止缓冲区溢出。但是被广泛使用的 C 语言却没有建立检测机制。标准 C 语言具有许多复制和添加字符串的函数，这使得标准 C 语言很难进行边界检查。C + + 略微好一些，但是仍然存在缓冲区溢出。一般情况下，覆盖其他数据区的数据是没有意义的，最多造成应用程序错误，但是，如果输入的数据是经过“黑客”或者病毒精心设计的，覆盖缓冲区的数据恰恰是“黑客”或者病毒的入侵程序代码，一旦多余字节被编译执行，“黑客”或者病毒就有可能为所欲为，获取系统的控制权。

溢出导致“黑客”病毒横行

缓冲区溢出是病毒编写者和特洛伊木马编写者偏爱使用的一种攻击方法。攻击者或者病毒善于在系统当中发现容易产生缓冲区溢出之处，运行特别程

序，获得优先级，指示计算机破坏文件，改变数据，泄露敏感信息，产生后门访问点，感染或者攻击其他计算机。

2000 年 7 月，微软 Outlook 以及 Outlook Express 被发现存在漏洞能够使攻击者仅通过发送邮件就能危及目标主机安全，只要邮件头部程序被运行，就会产生缓冲区溢出，并且触发恶意代码。2001 年 8 月，“红色代码”利用微软 IIS 漏洞产生缓冲区存溢出，成为攻击企业网络的“罪魁祸首”。2003 年 1 月，Slammer 蠕虫利用微软 SQL 漏洞产生缓冲区溢出对全球互联网产生冲击。而在近几天，一种名为“冲击波”的蠕虫病毒利用微软 RPC 远程调用存在的缓冲区漏洞对 Windows 2000/XP、Windows Server 2003 进行攻击，波及全球网络系统。据 CERT 安全小组称，*作系统中超过 50%的安全漏洞都是由内存溢出引起的，其中大多数与微软技术有关，这些与内存溢出相关的安全漏洞正在被越来越多的蠕虫病毒所利用。

缓冲区溢出是目前导致“黑客”型病毒横行的主要原因。从红色代码到 Slammer,再到日前爆发的“冲击波”，都是利用缓冲区溢出漏洞的典型。缓冲区溢出是一个编程问题，防止利用缓冲区溢出发起的攻击，关键在于程序开发者在开发程序时仔细检查溢出情况，不允许数据溢出缓冲区。此外，用户需要经常登录*作系统和应用程序提供商的网站，跟踪公布的系统漏洞，及时下载补丁程序，弥补系统漏洞。