旁注　蠕虫和病毒

蠕虫和病毒都试图在计算机中传播它们自己的代码段。正如 Spafford[105]所述，蠕虫(worm)可以自己运行，并且能够将自己的等效副本传播到其他机器。病毒(virus)能将自己添加到包括操作系统在内的其他程序中，但它不能独立运行。在一些大众媒体中，“病毒”用来指各种在系统间传播攻击代码的策略，所以你可能会听到人们把本来应该叫做“蠕虫”的东西称为“病毒”。

3.10.4　对抗缓冲区溢出攻击

缓冲区溢出攻击的普遍发生给计算机系统造成了许多的麻烦。现代的编译器和操作系统实现了很多机制，以避免遭受这样的攻击，限制入侵者通过缓冲区溢出攻击获得系统控制的方式。在本节中，我们会介绍一些 Linux 上最新 GCC 版本所提供的机制。

1. 栈随机化

为了在系统中插入攻击代码，攻击者既要插入代码，也要插入指向这段代码的指针，这个指针也是攻击字符串的一部分。产生这个指针需要知道这个字符串放置的栈地址。在过去，程序的栈地址非常容易预测。对于所有运行同样程序和操作系统版本的系统来说，在不同的机器之间，栈的位置是相当固定的。因此，如果攻击者可以确定一个常见的 Web 服务器所使用的栈空间，就可以设计一个在许多机器上都能实施的攻击。以传染病来打个比方，许多系统都容易受到同一种病毒的攻击，这种现象常被称作安全单一化(security monoculture)[96]。

栈随机化的思想使得栈的位置在程序每次运行时都有变化。因此，即使许多机器都运行同样的代码，它们的栈地址都是不同的。实现的方式是：程序开始时，在栈上分配一段 0～n 字节之间的随机大小的空间，例如，使用分配函数 `alloca` 在栈上分配指定字节数量的空间。程序不使用这段空间，但是它会导致程序每次执行时后续的栈位置发生了变化。分配的范围 n 必须足够大，才能获得足够多的栈地址变化，但是又要足够小，不至于浪费程序太多的空间。

下面的代码是一种确定“典型的”栈地址的方法：

```
int main() {
    long local;
    printf("local at %p\n", &local);
    return 0;
}
```

这段代码只是简单地打印出 `main` 函数中局部变量的地址。在 32 位 Linux 上运行这段代码 10 000 次，这个地址的变化范围为 `0xff7fc59c` 到 `0xffffd09c`，范围大小大约是 2^{23}。在更新一点儿的机器上运行 64 位 Linux，这个地址的变化范围为 `0x7fff0001b698` 到 `0x7fffffffaa4a8`，范围大小大约是 2^{32}。

在 Linux 系统中，栈随机化已经变成了标准行为。它是更大的一类技术中的一种，这类技术称为地址空间布局随机化(Address-Space Layout Randomization)，或者简称 ASLR[99]。采用 ASLR，每次运行时程序的不同部分，包括程序代码、库代码、栈、全局变量和堆数据，都会被加载到内存的不同区域。这就意味着在一台机器上运行一个程序，与在其他机器上运行同样的程序，它们的地址映射大相径庭。这样才能够对抗一些形式的攻击。