Shellcode

1.简介

在计算机安全中，shellcode是一小段代码，可以用于软件漏洞利用的载荷。被称为“shellcode”是因为它通常启动一个命令终端，攻击者可以通过这个终端控制受害的计算机，但是所有执行类似任务的代码片段都可以称作shellcode。Shellcode通常是以机器码形式编写的。

你可能想要利用已有的标准shellcode，比如来自Shell Storm数据库或由Metasploit的msfvenom工具生成。

2.Shellcode特点

（1）.字符串的直接偏移

即使你在C/C++代码中定义一个全局变量，一个取值为“Hello world”的字符串，或直接把该字符串作为参数传递给某个函数。但是，编译器会把字符串放置在一个特定的Section中（如.rdata或.data）。

由于需要与位置无关的代码，我们想把字符串作为代码的一部分，因此必须把字符串存储在栈上

（2）.函数地址

在C/C++中，调用某个函数非常简单。我们利用“#include<>”指定某个头文件，然后通过名称来调用某个函数。编译器和链接器会在背后帮你解决这个问题：解析函数的地址（例如，来自user32.dll的MessageBox函数），然后我们就可以轻而易举地通过名称调用函数。

在shellcode中，我们却不能以逸待劳了。因为我们无法确定包含所需函数的DLL文件是否已经加载到内存。受ASLR（地址空间布局随机化）机制的影响，系统不会每次都把DLL文件加载到相同地址上。而且，DLL文件可能随着Windows每次新发布的更新而发生变化，所以我们不能依赖DLL文件中某个特定的偏移。

我们需要把DLL文件加载到内存，然后直接通过shellcode查找所需要的函数。幸运的是，Windows API为我们提供了两个函数：LoadLibrary和GetProcAddress。我们可以使用这两个函数来查找函数的地址。

（3）.避免空字节

空字节（NULL）的取值为：0×00。在C/C++代码中，空字节被认为是字符串的结束符。正因如此，shellcode存在空字节可能会扰乱目标应用程序的功能，而我们的shellcode也可能无法正确地复制到内存中。

虽然不是强制的，但类似利用strcpy()函数触发缓冲区溢出的漏洞是非常常见的情况。该函数会逐字节拷贝字符串，直至遇到空字节。因此，如果shellcode包含空字节，strcpy函数便会在空字节处终止拷贝操作，引发栈上的shellcode不完整。正如你所料，shellcode当然也不会正常的运行。

3.Linux平台与Windows平台的shellcode对比

相对于Windows平台，编写针对Linux平台的Shellcode可能更为简单。这是因为在linux平台上，我们可以轻松地通过0×80中断执行类似write、execve或send的系统调用。

在linux平台上执行“Hello world”shellcode只需要以下几个步骤：

1. 指定系统调用syscall序号（如“write”）。
2. 指定系统调用syscall的参数（如，stdout，“Hellow， world”，字符串长度）

3）调用0x80中断来执行系统调用syscall。

这将会发起调用：write(stdout, “Hello, world”, length).

在Windows平台上，情况会更加复杂。为了生成更为可靠的shellcode，我们需要花费更多的步骤：

1）获取kernel32.dll 基地址；

2）定位 GetProcAddress函数的地址；

3）使用GetProcAddress确定 LoadLibrary函数的地址；

4）然后使用 LoadLibrary加载DLL文件（例如user32.dll）；

5）使用 GetProcAddress查找某个函数的地址（例如MessageBox）；

6）指定函数参数；

7）调用函数。