doi :10.3969/j.issn.1671-1122.2009.05.016

基于 Windows 平台的 RPC 缓冲区溢出漏洞研究

周虎生，文伟平

（北京大学软件与微电子学院信息安全系，北京 102600）

摘 要：Windows系统中潜在的RPC缓冲区溢出漏洞是目前信息系统面临的最严重安全威胁之一。本文介绍了RPC原理，分析了缓冲区溢出的原理，总结了RPC漏洞分析的一般方法，并针对一个RPC漏洞进行了详细分析，包括定位溢出函数，分析溢出点，溯源和溢出实现。

关键词：RPC；缓冲区溢出；漏洞

中图分类号：TP393.08 文献标识码：A

The Research of RPC Overflow Vulnerability

ZHOU Hu-sheng, WEN Wei-ping

(Department of Information Security, SSM, Peking University, Beijing 102600*, China)*

Abstract: The RPC overflow vulnerability existing in Windows operating system constitute one of the most serious security threat towards computer network. In this paper the principles of RPC are introduced, theories of buffer overflow are analyzed and the methods of analyzing the RPC overflow vulnerability are also summarized. In allusion to a vulnerability it analyzes the process of the RPC overflow vulnerability exploit, including locking overflow function, analyzing overflow point and achieving exploit.

Key words: RPC;buffer overflow;leak

0 引言

随着计算机网络技术的高速发展，人们的生活对计算机和网络的依赖程度越来越高。与此同时，各种网络攻击行为也越来越频繁，而操作系统和软件中的各种漏洞为攻击者大开方便之门。其中缓冲区溢出漏洞已经成为当今计算机系统中最具威胁的漏洞，RPC（Remote Procedure Call）缓冲区溢出漏洞就是其中的一种。

Windows 系统中，为了方便分布式操作，广泛应用了RPC，如局域网中的文件共享，打印机共享，分布式数据库。因此对 RPC 缓冲区溢出漏洞的研究是十分必要的。

1 RPC 的基本原理

RPC 是操作系统的一种远程过程调用协议，是分布式计算中常用到的技术。目前，使用最普遍的是开放式软件基础的分布式计算环境（DCE）[1]，微软所用的 RPC 系统就是基于 DCE RPC 开发的。计算机间通讯的过程分为两种形式：一种是数据的交换，一种是进程的通讯，RPC 属于后者。RPC 可以让程序员调用一个函数，而这个函数是在另外一台或多台计算机上运行的，执行完后将结果传回本机进行后续操作。调用过程中的网络操作对程序员是透明的，在代码中调用一个远程函数就像调用本地的一样方便，只要把接口定义好，RPC 体系会帮助完成建立网络链接、会话握手、身份验证、参数传递、返回结果等细节操作。

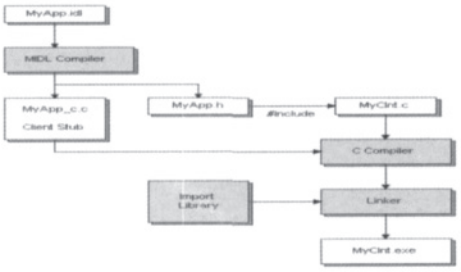

图1 RPC客户端开发流程（引自MSDN）

在 RPC 调用前，会产生一个包含一个提示符的 IDL（Interface Definition Language）原型文件 [2][3]，然后对编辑好的文件进行编译，生成三个文件 [3]：客户端 stub，服务器端 stub，RPC 调用的头文件。其中 stub 负责网络操作过程中的所有细节。IDL 中包含定义接口用的头，指明 UUID 和 endpoint，用来唯一指明一个接口。

2 缓冲区结构及溢出原理

常见的缓冲区溢出包括栈溢出和堆溢出，下面以 Windows 为例分别进行介绍。

2.1 栈结构及溢出原理

当一个函数调用另一个函数的时候，其栈帧结构如图 2 所示。在正常情况下，被调函数执行完毕，将被调函数的返回地址传给 EIP 寄存器，CPU 从 EIP 寄存器取指令地址，继

续执行主函数。被调函数的局部变量在栈中一个挨着一个排列，如果这些局部变量中有数组之类缓冲区，并且程序中存在数组越界的缺陷，那么越界的数组元素就有可能破坏栈中的相邻变量的值，甚至破坏栈帧中所保存的EBP值，EIP值等重要数据[4][5]。成功的缓冲区溢出攻击会用精心设计的数据覆盖返回地址，使被调函数执行完后将修改后的返回地址传入EIP，从而使CPU执行攻击代码（shellcode）。

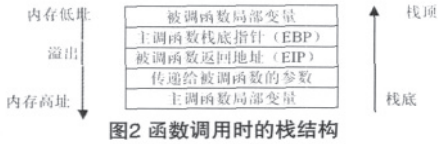

图2 函数调用时的栈结构

2.2 堆结构及溢出原理

堆区的内存按不同大小组织成块，以堆块为单位进行标识。一个堆块包含两部分：块首和块身。块首是一个堆块头部的几个字节，用来标识这个堆块自身的信息；块身是紧跟在块首后面的部分，也是最终分配给用户使用的数据区。

堆块分为占用态的堆块和空闲态的堆块，如图3所示，他们在结构上的区别是空闲态的堆块在块首增加了两个指针Flink和Blink，分别指向前一个和后一个同样大小的空闲堆块。占用态的堆块被使用它的程序索引，空闲态的堆块被堆表索引。其中，最重要的堆表有两种：空闲双向链表Freelist和快速单向链表Lookaside。

Freelist是一个128双链表的数组，2到127单元大小的空闲堆块根据他们的大小存在链表中，其中Freelist[0]用来存放超过127单元大小的堆块；Lookaside是一个128单向块链表，目的是为了加快小块（小于127个分配单元，即1016字节）的分配速度，堆在分配和释放堆块时总是优先使用Lookaside表，失败时才用Freelist表[3][4][5]。

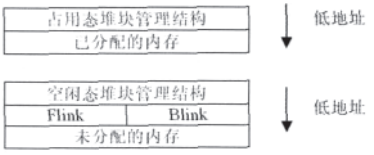

图3 堆块结构

堆管理系统的三类操作：堆块分配、堆块释放、和堆块合并，都是对链表的操作，我们可以通过伪造链表结点的指针，在堆块释放和合并的过程中获得一次读写内存的机会。对溢出利用的精髓就是用精心构造的数据去溢出下一个堆块的块首，改写块首中的Flink和Blink，然后在分配、释放、合并等操作发生时伺机获得一次向内存任意地址写入任何数据的机会[3][5]。

3 RPC漏洞实例分析

MS06-040是微软公布的一个漏洞，Ms06-040是微软的编号，其CVE编号为CVE-2006-3439，对应补丁编号为KB921883，微软对它的描述是：Server 服务中存在一个远程执行代码漏洞，成功利用此漏洞的攻击者可以完全控制受影响的系统。 同其他缓冲区溢出漏洞一样，分析该溢出漏洞的关键是定位溢出函数，寻找关键溢出点，计算跳转地址[6]。

本文以Ms06-040为例，操作系统为Windows 2000，介绍RPC缓冲区溢出漏洞的分析方法。

3.1 定位溢出函数

若机器打过补丁，可以在\\Winnt \$NtUninstallKB921883$目录下找到补丁前的netapi32.dll。对补丁前后的dll文件进行反汇编分析，使用补丁比对工具，如Eeye的EBDS和Darun Grim，Bindiff等，可以很快查到微软在补丁中修改的地方，查看一下一个改动较大的函数，sub_7517FC68，若下载了微软的符号文件，就能显示出该函数的名字是CanonicalizePathName。使用IDA可以查看该函数的调用关系图，选择chart of xrefs to功能，看到是NetpwPathCanonicalize调用了CanonicalizePathName这个子函数；选择chart of xrefs from功能，看到CanonicalizePathName函数调用了__imp_wcscat，__imp_wcscpy等容易发生溢出的系统函数。至此，可以将溢出函数定位缩小到NetpwPathCanonicalize及其调用的子函数CanonicalizePathName上。

3.2 分析溢出点

查看问题函数NetpwPathCanonicalize在IDA中反汇编出的代码[5][7]：

```
.text:7517F7E2 ; Exported entry 303. NetpwPathCanonicalize
.text:7517F7E2 ; int __stdcall NetpwPathCanonicalize(wchar_t *lpWideCharStr, int, int, wchar_t *Source, int, int)
```

得出结论，该函数是netapi32.dll的导出函数(Exported)，有6个参数。后面调用CanonilizePathName，之前进行了一些参数检查，在这次验证中，如果第4个参数的Unicode长度超过0x103则引起程序退出。在CanonicalizePathName中，实现的功能是将NetpwPathCanonicalize函数的第4个参数所指的字符串连接上‘\’和第1个参数所指的字符串，这个函数使用局部变量，在栈内开空间暂存新串，这块空间可以溢出。由于前面对4号串的长度进行了检查，所以无法仅仅由4号串实现溢出，但是后面忽略了对1号串的检查，使得在使用wcscpy连接1号串的时候发生溢出成为可能，1号串足够长的话可以覆盖掉EBP和EIP。

3.3 溯源

RPC通常是以opnum来确定函数的接口，在得知某个函数（opnum）可以触发溢出时，攻击者可以构造出触发这个函数的数据包来触发漏洞。NetpwPathCanonicalize这个函数可以被RPC远程调用，下面我们找一下通过谁可以调用它，查询《Windows network services internals》[8]，可以查到以下信息：

```
The srvsvc interface is used to manage the lanmanserver service.
Interface Operation number Operation name
4b324fc8-1670-01d3-1278-5a47bf6ee188 v3.0: srvsvc0x00 NetrCharDevEnum
0x01 NetrCharDevGetInfo
```

```
0x02 NetrCharDevControl
0x03 NetrCharDevQEnum
……
0x1e NetprPathType
```

0x1f NetprPathCanonicalize//这个就是我们想要找的调用的地方了

```
0x20 NetprPathCompare
```

我们要找的接口在srvsvc.dll中的opnum为0x1f的地方。利用IDA反汇编srvsvc.dll，并使用一个极好的抽象RPC接口的IDA插件mIDA，寻找opnum为0x1f的函数，跟进去可以确定就是这里调用的netapi32.dll中的NetpwPathCanonicalize：

```
.idata:76781260    extrn __imp_NetpwPathCanonicalize:dword。
```

可以得到接口定义：

```
[ uuid(4b324fc8-1670-01d3-1278-5a47bf6ee188),
 version(3.0) ]
interface mIDA_interface
{/* opcode: 0x1F, address: 0x767897B4 */
     long  sub_767897B4 (
 [in][unique][string] wchar_t * arg_1,
 [in][string] wchar_t * arg_2,
 [out][size_is(arg_4) ] char * arg_3,
 [in] long arg_4,
 [in][string] wchar_t * arg_5,
 [in, out] long * arg_6,
 [in] long arg_7
) ;
}
```

IDL中定义的接口比NetpwPathCanonicalize多一个参数，这个是RPC加上去的，只要不指向空，实际中不会传递给netapi32.dll里的NetpwPathCanonicalize。

RPC体系在向远程接口映射具体函数的时候，是按照IDL里面函数定义的顺序来定位的，而不是函数的名称。例如此例中NetprPathCanonicalize是第0x1f个函数，我们要想准确调用，就要在前面定义0x1e个函数。

3.4 溢出实现

本例先通过一个简单的栈溢出重现溢出过程，本地溢出POC(Proof Of Concept)主要代码[5][7]：

```
char arg_1[0x320];
char arg_2[0x440];
int arg_3=0x440;
char arg_4[0x100];
int arg_5=44;
Trigger = (MYPROC) GetProcAddress(LibHandle, VulFunc) ;
memset(arg_1,0,sizeof(arg_1)) ;
memset(arg_1,'a',sizeof(arg_1) -2) ;
memset(arg_4,0,sizeof(arg_4)) ;
memset(arg_4,'b',sizeof(arg_4) -2) ;
(Trigger) (arg_1,arg_2,arg_3,arg_4,&arg_5,0)
```

出现错误之后调试，可以得出栈中的数据分布，大致如图4：

图4 溢出发生时栈中的布局

可以定位出arg_1的第0x318到0x31B的四个字节的数据可以覆盖返回地址。

在远程溢出的时候，经过一系列管道、接口的设置之后，就可以像调用本地动态链接库一样调用远程主机。为了增强程序的健壮性，我们使用具有JMP ESP功能的跳板地址实现，这里使用的通用地址是0x7ffa4512。远程溢出的缓冲区布局如图5所示：

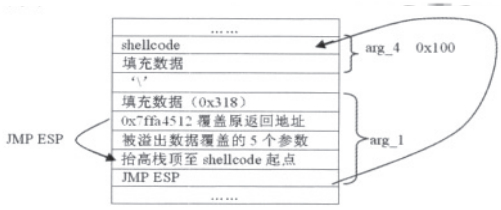

图5 远程溢出的缓冲区布局示意图

按照上述思路，构造远程溢出代码，主要部分如下：

```
memset(arg_1,0,sizeof(arg_1)) ;
memset(arg_1,'a',sizeof(arg_1) -2) ;
memset(arg_4,0,sizeof(arg_4)) ;
memset(arg_4,'b',sizeof(arg_4) -2) ;
memcpy(arg_4,shellcode,sizeof(shellcode)) ;
arg_1[0x318]=0x12;// 跳板的地址
arg_1[0x319]=0x45;
arg_1[0x31A]=0xfa;
        arg_1[0x31B]=0x7f;
```

用上述思路构造出的远程溢出shellcode对Windows2000操作系统的靶机进行溢出，shellcode得以执行，达到预期结果。

4 结束语

由于RPC在网络中使用的普遍性，RPC漏洞对网络安全的威胁极大，必须提高安全意识，进一步研究漏洞的利用方法，从根本上预防此种漏洞的出现和针对此种漏洞的攻击，增强系统的安全性。　　（责编　潘静）

参考文献：
[1] DCE 1.1. Remote Procedure Call [EB/OL]. http://www.opengroup.org/public/pubs/catalog/c706.htm,15/08/1994
[2] Microsoft Corporation. MSDN. http://msdn.microsoft.com/zh-cn/default.aspx. 2005
[3] 孙晓妍,武东英,季明,郭宁. Windows系统下RPC堆溢出的研究,微电子学与计算机，2007 Vol.24 No.6:170-172.
[4] 许俊杰,蔡皖东. 一种缓冲区溢出漏洞检测模型及系统实现. 计算机科学,2008,Vol.35,No.6:60-62.
[5] 王清. 0day：软件漏洞分析技术，北京：电子工业出版社,2008.38-179.
[6] Friddy. RPC漏洞的通用分析方法,http://www.friddy.cn/article.asp?id=68,2009-01-07.
[7] Inject2006. 深入浅出MS06-040. http://blog.csdn.net/inject2006/archive/2008/08/31/2854984.aspx,2008-08-31
[8] Jean-Baptiste Marchand. Windows network services internals. http://hsc.fr/ressources/articles/win_net_srv/index.html,2006.

作者简介：周虎生(1986－)，男，硕士研究生，主要研究方向：系统与网络安全；文伟平(1976－)，男，副教授，主要研究方向：网络攻击与防范、恶意代码研究、信息系统逆向工程和可信计算技术等。