纵横互联网系列丛书

黑客帝国

Internet安全防范实例

冯晓聪
王腾宇
梅珊 编著
吴斌 监制
程峰 策划

本书附赠光盘1张

中国电力出版社
www.infopower.com.cn

纵横互联网系列丛书

黑客帝国

Internet安全防范实例

冯晓聪 王腾宇 梅珊 编著

吴斌 监制

程峰 策划

中国电力出版社

内 容 提 要

本书是纵横互联网系列丛书之一。

本书详细介绍了利用 Winsock 网络编程接口及 Visual C++编程实现的几种网络攻击的基本原理。书中精解了十二个网络攻防实例，通过实例读者可以了解黑客、了解黑客的攻击原理、了解网络编程，真正做到纵横互联网，“知己知彼，百战不殆”。

本书讲解全面细致，举例典型实用，适合初级读者，并可作为具有相当水平的中、高级编程设计人员的参考书。

中国电力出版社出版、发行

（北京三里河路 6 号　100044　http://www.infopower.com.cn）

三河市实验小学印刷厂印刷

各地新华书店经售

*

ISBN 7-900038-92-2/TP・79

2002 年 5 月第一版　2002 年 5 月北京第一次印刷

787 毫米×1092 毫米　16 开本　17.5 印张　429 千字

定价 30.00 元

楔　子

——Internet 随笔

我是天空里的一片云，

偶尔投影在你的波心——

　　你不必讶异，

　　更无须欢喜——

在转瞬间消灭了踪影。

你我相逢在黑夜的海上，

你有你的，我有我的，方向；

　　你记得也好，

　　最好你忘掉，

在这交会时互放的光亮！

——摘自徐志摩的诗《偶然》

当我很小的时候，没想过未来的美好……

当我初入学堂的时候，没想过以后的人生之路怎么走……

当我第一次上网的时候，发现了一个全新的世界……

虚拟世界的大门敞开了，我将走进去，从此开始我的第二次生命……

万籁俱寂，时间停止了，此刻即是永远……，永远也即是此刻……

冬日里，那条布满金黄色树叶的大街披上了茫茫的白雪，而那些随着时光的流逝如今已发黄了的老照片和我那相依如命的破旧牛仔裤都没有因新千年的到来而改变。正如苏东坡所言："归去，也无风雨也无晴"，道尽了 20 世纪末网络的兴衰。匆匆几个过客谁又能写尽 Internet 的沧桑，也许下笔时，一切都已经事过境迁，今非昔比。

我们就像茫茫夜空中的繁星，各自沿着自己的轨道前行，彼此相隔甚远，偶尔擦肩而过，谁也不知道以后会不会再相见，以及以怎样的方式相见。如果说一切都出于偶然，那么这个偶然也必是源于几亿万年前的某个缘份吧。无论日后我们的轨道会否重叠，当初擦肩而过的瞬间互放的光亮应该是每一个斑驳回忆中惟一不会褪色的美丽吧。有人说，美好的事物是无法长久的，它所有的美丽会在悲怆中完结。也许是吧，但也许会留下回忆，只是这回忆却是酸楚的。

时光荏苒，岁月消逝，年轻的心都飘尽了。近来网绪茫茫，每一次坐在电脑前，在惆怅的音乐氛围中，静听敲击键盘的声响，如雨打残荷，不禁感慨万千。

自从迷上网络，就仿佛踏上一条不归之路。万籁俱寂，空气似乎也凝固了。下雨时，绿树梢头浮起一片片白光，隐约可见。那时我似乎还不以雨中跋涉为苦，虽淋湿了衣服，但还怀着微妙的心情，伫立在梦桥之上，俯瞰潺潺的流水，凝视着水面上泛起的涟漪，这时，会更加感受到雨的亲切和洒脱。烟雨濛濛，轻纱似地笼住一切，淡极欲无。风雨网络，发乎自然。

而今点击鼠标已成为一种习惯，已清楚地知道在网络的密林中哪里会有浆果，那里会有清泉和木屋，一批批网友轻轻地纷至沓来，然后又轻轻地远去，当所有已实践的、未实践的网络情缘全部消失在记忆的边缘时，屏幕后面模糊了一张张未知的笑脸，网络给予了找寻失去的梦想的希望。纵有迷失和惊喜无数，但什么也阻挡不了渔人对海的热爱。

有人告诉我，江湖险恶！

也有人告诉我：江湖精彩！

但我说：既然春天已经到来，那就让我们去闯荡江湖吧！

好了，要说的都写在了文章里，而没有说出来的也都随风而逝了……。

丛书前言

网络的诞生，改变了人们使用计算机的方式；而 Internet 的出现，又改变了人们使用网络的方式。Internet 的快速发展，代表着人类历史上另一场革命的开始。曾有人断言：假如有一天，21 世纪的人类文明就像玛雅文化一样神奇地消失的话，那么，当后世的考古学家在发掘这段历史的时候，一定会把 Internet 当做是一场瘟疫。因为，它在极短的时间内充斥了各种各样的媒体。虽然，这只是笑谈，但从中不难看出 Internet 对人类的生活所造成的影响是何等深远而且广泛！

网络时代已经来临！缤纷多彩的网络世界改变着人们的生活，特别是当 WWW 日益普及之际。人们对信息资源的渴求是无止境的，开发者们则在不断地扩充超媒体语言的能力。从静态文本到静态图像，从静态图像再到动态图像，随后又加入了声音、影像、三维动画等，网络世界从此变得生机盎然。

目前，图书市场上出现了很多介绍 Internet 编程方面的书籍。但是，这些书籍或者过于简单，让读者觉得收获甚微；或者因过于专业化而未能兼顾初级水平的用户；或者着重于理论，而缺乏具体的实例剖析；或者理论与实例相背离，没有起到应有的效果。因考虑到很多用户并没有太多的实际编程经验，故本丛书力图通过实例，系统全面地介绍网络编程各个方面的基本原理和开发技巧，以尽可能地满足不同层次读者的要求，使读者能通过本丛书切实掌握一门网络编程语言的技巧和方法，可以开发出自己功能强大的网站。

本丛书以讲解和分析实例为主。丛书中的每本书籍，包括从最简单的例子到高级的编程技巧，都有所涉猎。希望通过学习本丛书，无论是网络开发的新手，还是经验丰富的“老鸟”，都可以从中获益。

本丛书的讲解全面细致，举例典型实用，使读者可以迅速走入网络编程的一片新的天地，继而可以熟练地使用一门网络编程语言进行开发。同时，本丛书为专业的开发人员提供了详尽的参考，有利于进一步提高编程的水平，掌握更科学的编程技巧。本丛书的实例都是按照众多高手们从入门到精通的学习过程来编排的，以方便读者循序渐进地学习。总之，编写本丛书的目的就是为广大读者提供一套系统全面的网络编程的实用教程，希望本丛书的出版可以对大家的实际工作有所帮助。

编写本丛书可不是一件轻松的事情，其中凝聚了太多人的努力和无私的奉献。首先，要感谢每一本书的作者，本丛书是他们渊博知识的凝聚，也是他们心血的结晶。还要感谢编校人员，他们认真细致的工作作风使得本丛书尽量少出错。最后还要感谢在我身后一直给予支持的家人和朋友们，有了你们的理解和支持，这栋大厦才得以构建。

对于本丛书中的每一本书，由于每位作者的学识、能力有限，书中疏漏或错误之处在所难免，敬请广大读者批评指正。

前　言

随着时代的变迁，Internet 逐渐为人们所熟悉。越来越多的人奔向这片广阔而自由的海洋，休闲、购物、学习、交友，尽情享受冲浪的欢乐。网络的营运已成为了社会的新时尚。然而 Internet 在方便信息交流的同时，也为人们提出了一个新的课题——网络安全吗？

或许有些人已经意识到了网络安全的重要性，并采取了必要的手段进行了自我保护。但仍然有很多的人丝毫没有安全意识，甚至连什么是防火墙，什么是黑客技术都不了解，这是非常令人担心的现象。作为新时代的弄潮儿，要有的不仅仅局限于会用计算机、会上网，更要有基本的网络知识和高度的安全意识！要知道，我们面对的 Internet 是如此得庞大，也是如此得脆弱。有一群神秘的“黑衣人”，通过所谓讳莫如深的技术，穿梭于网际，说不定明天就会咚咚地扣响你的计算机的大门。本书的编写正是出于这样一个目的，让更多的人了解黑客，了解网络编程，让读者从一个个精辟的实例了解黑客攻击的原理，从而做到“知己知彼，百战不殆”！

本书详细介绍了几种网络攻击实现的基本原理、Winsock 网络编程接口及 Visual C++ 的具体编程实现。虽然主要面向初级读者，对于具有相当水平的中、高级编程设计人员，本书也同样是一本有价值的参考书。

本书共精解了十二个网络攻防实例。

实例一：获取本机 IP 地址。介绍了如何使用 Winsock 提供的 API 函数和 Visual C++ 获取本机所设置的 IP 地址。IP 攻击作为黑客惯用的伎俩，利用的是最基本的网络技术。

实例二与实例三：讨论了客户与服务器方式的网络通信。在这两个实例中，介绍了网络通信基于的各种协议，并详细介绍了 Windows 套接字——Winsock。

实例四：两台机器间的串口通信。在某些情况下，我们不需要借助于网络，用串行线连接两台机器直接进行通信是一种更加容易的方式。在这个实例中，介绍了如何在用串行线相连的两机之间实现通信。

实例五：多机通信。本实例通过一个简单的聊天室程序，详细介绍了在一个局域网内多台计算机之间通信的原理。

实例六：洪水 Ping 攻击。Ping 命令可以用来确定一个指定的主机位置。在实际的使用过程中，黑客常常会利用 Ping 协议或者其他的公开协议，来收集驻留在网络系统中的各个主机系统的相关信息。本实例介绍了如何利用最少资源的 Ping 命令，攻击目的主机，使之消耗最大资源而死机。

实例七：网络远程控制。本实例将让大家真真切切感受黑客的味道，锁定远程计算机，获取屏幕图像，控制鼠标动作等等。

实例八：利用“侦听-转发”程序破译网管协议。也许很多人对网络管理的概念还不熟悉，通过本实例，可以清楚地了解网络管理程序、网络设备及其通信协议的概念。并通过具体的程序设计，实现截取网管程序与被管设备之间通信的信息。

实例九：如何获取网卡 MAC 地址。MAC 地址，简单地说就是网卡的物理地址。它在

实际应用中扮演着很重要的角色，如与 IP 地址绑定、限制连接 Internet 等。本实例将使读者掌握如何获取网卡的 MAC 地址。

实例十：网络漏洞分析。虽然越来越多的人已经认识到了网络安全的重要性，但在实际应用中，不管是操作系统还是各种软件都存在不同的安全漏洞，而黑客正是利用这些漏洞对目标主机进行攻击的。本实例针对一个特定的实例——guestbook.cgi 中的漏洞进行分析，并编程实现利用该漏洞进行的网络攻击。

实例十一：截获网络数据报。数据报中包含了很多有价值的信息，所以截获数据报也是黑客进行窃听的重要手段。本实例实现了经常用在截获网络数据报中的一个 sniffer 类。

实例十二：Proxy 服务器。代理服务器是实现多个用户共享一条线路上网必不可少的软件。在本实例中将讲述如何实现代理服务器，以达到多台电脑共享上网的目的。

相信通过学习这十二个实例，读者能够掌握网络通信的基本概念，了解各种协议，熟悉 Visual C++ 集成化编程环境，理解 Winsock 库中的重要函数，并对黑客技术有了一种基本的认识。笔者给出的实例只是起一个抛砖引玉的作用，希望各位读者能发挥自己的聪明才智，编写出更出色的“黑客”程序。本书搜集的十二个实例的示范代码也同时包括在配套光盘中。请各位读者亲自实践。

在本书的编书过程中，李洪源同学提供了很大的帮助，并提出了很多宝贵意见，在这里向他表示最诚挚的感谢！

由于黑客技术博大精深，加之笔者水平有限，时间仓促，书中错误、遗漏之处在所难免，敬请广大专家、读者批评指正。

作　者

目　　录

实例1 获取本机IP地址

都说“Free”（自由、免费）是 Internet 的灵魂，我们可以很轻松地链接到喜爱的站点浏览新闻、下载软件，但是同时，其他人，例如“黑客”也能够很方便地连接到你的机器。实际上，大多数拨号上网的家庭用户的电脑都基本是不设防的，糟糕的安全防范设置无意间在机器和系统中留下了大量的“后门”，也就相当于给黑客打开了大门。这决不是危言耸听，当笔者到许多朋友家中的时候，竟然发现他们经常上网的电脑就连基本的防火墙都没有安装，更不用说其他更高级的设置了。

了解黑客，是为了更好地防范互联网上的安全攻击；了解网络编程，是为了更加深入地了解黑客攻击的原理。所以，本书的宗旨是在介绍黑客知识的同时，重点讲解网络编程的实例，通过实例的讲解，使读者对于黑客有一个较高层次的认识。

什么，你还不了解“黑客”？还不知道黑客的常用攻击手段和防范措施？请关注本实例，在实现了本实例的功能后，我们将向初级读者简单介绍黑客的常用攻击手段和防范措施。

☆ 实例目标

网络安全方面的开发工作是一项有意义而又比较艰辛的工作。在网络已经普及到千家万户的今天，懂得一些必要的网络安全知识，对于大多数程序员（包括笔者在内），都是十分必要的。网络安全知识，使得程序员明白网络数据传输及存储到底是怎么一回事，以及如何进行加密保护，让大多数人懂得网络安全意识的重要性，以保护自己的电脑不受伤害。

可以这么说：上网的时间越多，被别人通过网络侵入机器的可能性也就越大。如果黑客们在你的设置中发现了安全方面的漏洞，就会对你发起攻击。这种攻击可能是一般的骚扰，如降低你的速度或者让你的机器崩溃，也可能会很严重，例如浏览本人不想让他人知道的信息、偷窃口令和其他敏感资料。

但是很多人并不以为然，特别是一些拨号上网的用户，他们有这样的一种错误认识：我是用拨号上网，所以我的机器是安全的。

的确，对于拨号上网的用户来说，每次使用的 IP 地址都是不同的，也即动态 IP，所以相比静态 IP 的用户而言，黑客是很难找到，但是有一些扫描手段的黑客软件可以在 1 个小时以内逐个扫描上万个 IP 地址，这是非常可怕的，只要黑客使用了这些工具，即使是拨号上网的用户也可能受到攻击。那么，你能通过编程获取你的机器的 IP 地址吗？

互联网安全防范方面的编程比较复杂，我们就从获取 IP 地址这个最简单的例子开始吧！

在本实例中，介绍了如何使用 Visual C++获取本机所设置的 IP 地址，只有懂得了像 IP 地址这样的初步网络知识，才能了解网络传输的基础，从而对网络知识有一些初步的了解。这些知识为你将来在网络方面进行更深入地工作奠定基础。

实现技术

本实例讲述的是在 Win32 平台上的 Winsock 编程。对于众多的基层网络协议，Winsock 是访问它们的首选接口。而且在每个 Win32 平台上，Winsock 都以不同的形式存在着。Winsock 是网络编程接口，而不是协议。

既然我们采用了目前对于 Windows 平台的最流行的 Windows Sockets 规范来进行网络编程，那么，究竟什么是 Windows Sockets 规范呢？Windows Sockets 规范以 U.C. Berkeley 大学 BSD UNIX 中流行的 Socket 接口为范例，定义了一套 Microsoft Windows 下网络编程接口。它不仅包含了人们所熟悉的 Berkeley Socket 风格的库函数，也包含了一组针对 Windows 的扩展库函数，以使程序员能充分地利用 Windows 消息驱动机制进行编程。Windows Sockets 规范就是建立在 Bekeley 套接口模型上的。Bekeley 套接口模型现在已是 TCP/IP 网络的标准。它提供了习惯于 UNIX 套接口编程的程序员极为熟悉的环境，并且简化了移植现有的基于套接口的应用程序源代码的工作。

Windows Sockets 规范本意在于提供给应用程序开发者一套简单的 API，并让各家网络软件供应商共同遵守。此外，在一个特定版本 Windows 的基础上，Windows Sockets 也定义了一个二进制接口（ABI），以此来保证应用 Windows Sockets API 的应用程序能够在任何网络软件供应商的符合 Windows Sockets 协议的实现上工作。因此这份规范定义了应用程序开发者能够使用，并且网络软件供应商也能够实现的一套库函数调用和相关语义。

遵守这套 Windows Sockets 规范的网络软件，我们称之为 Windows Sockets 兼容，而 Windows Sockets 兼容实现的提供者，我们称之为 Windows Sockets 提供者。一个网络软件供应商必须百分之百地实现 Windows Sockets 规范，才能做到 Windows Sockets 兼容。

任何能够与 Windows Sockets 兼容实现协同工作的应用程序就被认为是具有 Windows Sockets 接口。我们称这种应用程序为 Windows Sockets 应用程序。

Windows Sockets 规范定义并记录了如何使用 API 与 Internet 协议族（IPS，通常我们指的是 TCP/IP）连接，尤其要指出的是所有的 Windows Sockets 实现都支持流套接口和数据报套接口。

现有的一套 Windows Sockets API 能够在所有 3.0 以上版本的 Windows 和所有 Windows Scokets 实现上使用，所以它不仅为 Windwos Sockets 实现和 Windows Sockets 应用程序提供了 16 位操作环境，而且也提供了 32 位操作环境。并且，Windows Sockets 也支持多线程的 Windows 进程。一个进程包含了一个或多个同时执行的线程。

应用程序调用 Windows Sockets 的 API 实现相互之间的通信。Windows Sockets 又利用下层的网络通信协议功能和操作系统调用实现实际的通信工作。它们之间的关系如图 1-1 所示。

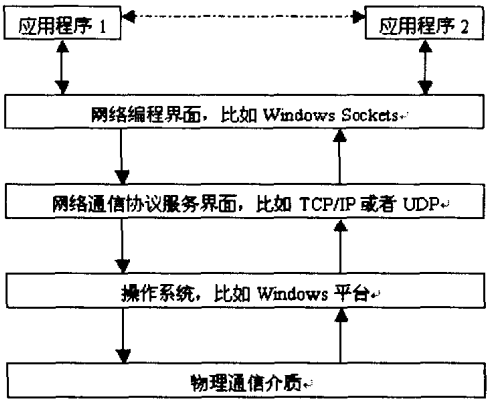

图 1-1　应用程序与 Windows Sockets 关系示意图

具体地来说，本实例所涉及的主要技术如下。

使用 WSAStartup 函数

Winsock 初始化的时候，每个 Winsock 应用都必须加载 Winsock DLL 的相应版本。如果调用 Winsock 之前，没有加载 Winsock 库，这个函数就会返回一个 SOCKET_ERROR 的错误消息，错误信息是 WSANOTINITIALISED。加载 Winsock 库是通过调用 WSAStartup 函数实现的。这个函数的原型如下：

```
int WSAStartup
(
  WORD wVersionRequested,
  LPWSADATA lpWSAData
);
```

其中，参数的说明如下：

- wVersionRequested：指定准备加载的 Winsock 库的版本。高位字节指定所需要的 Winsock 库的副版本，而低位字节则是主版本。然后，可用宏 MAKEWORD(X,Y)（其中，X 是高位字节，Y 是低位字节）方便地获得 wVersionRequested 的正确值。
- lpWSAData：是指向 LPWSADATA 结构的指针，WSAStartup 用其加载的库版本的有关信息填在这个 lpWSAData 结构中。

WSAStartup 函数必须是应用程序或 DLL 调用的第一个 Windows Sockets 函数。它允许应用程序或 DLL 指明 Windows Sockets API 的版本号及获得特定 Windows Sockets 实现的细节。应用程序或 DLL 只能在一次成功的 WSAStartup()调用之后才能调用进一步的 Windows Sockets API 函数。

表 1-1 列出了各种微软 Windows 平台支持的 Winsock 最新版本。需要记住的重点是各个版本之间的差别。Winsock 1.X 不支持多数高级 Winsock 特性。除此以外，对采用 Winsock 1.0 的应用而言，必须有 Winsock.H 包含文件，而对使用 Winsock 2 的应用而言，则需要 Winsock 2.H 包含文件。

表 1-1 不同版本的平台支持的 Winsock 版本

平 台	Winsock 版 本
Windows 95	1.1(2.2)
Windows 98	2.2
Windows NT	2.2
Windows 2000	2.2
Windows CE	1.1

为支持日后可能和 Windows Sockets 1.1 有功能上差异的 Windows Sockets 实现及应用程序，在 WSAStartup()中规定了一个协议，WSAStartup()的调用方和 Windows Sockets DLL 互相通知对方各自可以支持的最高版本，并且互相确认对方的最高版本是否可接受。在 WSAStartup()函数的入口，Windows Sockets DLL 检查应用程序所需的版本，如果版本高于 DLL 支持的最低版本，则调用成功。并且 DLL 在 wHighVersion 中返回它所支持的最高版本，在 wVersion 中返回它的高版本和 wVersionRequested 中的较小者。然后 Windows Sockets DLL 就会假设应用程序将使用 wVersion。如果 WSADATA 结构中的 wVersion 域对调用方来说不可接收，它就应调用 WSACleanup()函数，并且要么去另一个 Windows Sockets DLL 中搜索，要么初始化失败。

本协议允许 Windows Sockets DLL 和 Windows Sockets 应用程序共同支持一定范围的 Windows Sockets 版本。如果版本范围有重叠，应用程序就可以成功地使用 Windows Sockets DLL。表 1-2 给出了 WSAStartup()在不同的应用程序和 Windows Sockets DLL 版本中是如何工作的。

表 1-2 应用程序和 Windows Sockets DLL 版本协调工作的关系

应用程序版本	DLL 版本	wVersionRequested	wVersion	wHighVersion	最 终 结 果
1.1	1.1	1.1	1.1	1.1	use1.1
1.0，1.1	1.0	1.1	1.0	1.0	use1.0
1.0	1.0，1.1	1.1	1.1	1.1	use1.0
1.1	1.0，1.1	1.1	1.1	1.1	use1.1
1.1	1.0	1.1	1.0	1.0	失败
1.0	1.1	1.0	—	—	WSAVERNOTSUPPORTED
1.0，1.1	1.0，1.1	1.1	1.1	1.1	use1.1
1.1，2.0	1.1	2.0	1.1	1.1	use1.1
2.0	1.1	2.0	1.1	1.1	失败

一旦应用程序或 DLL 进行了一次成功的 WSAStartup()调用，它就可以继续进行其他所需的 Windows Sockets API 调用。当它完成了使用该 Windows Sockets DLL 的服务后，应用程序或 DLL 必须调用 WSACleanup()，以允许 Windows Sockets DLL 释放任何该应用程序的资源。

实际的 Windows Sockets 实现细节在 WSAData 结构中描述如下：

```
typedef struct WSAData
{
    WORD                    wVersion;
    WORD                    wHighVersion;
    Char                    szDescription[WSADESCRIPTION_LEN+1];
    Char                    szSystemStatus[WSASYS_STATUS_LEN+1];
    unsigned short          iMaxSockets;
    unsigned short          iMaxUdpDg;
    char FAR *              lpVendorInfo;
}
WSADATA, FAR * LPWSADATA;
```

WSAStartup()函数把 WSAData 结构的第 1 个字段 wVersion 设成打算使用的 Winsock 版本。WHighVersion 参数容纳的是现有的 Winsock 库的最高版本。请读者记住，在这两个字段中，高位字节代表的是 Winsock 副版本，而低位字节代表的则是 Winsock 主版本。szDescription 和 szSystemStatus 这两个字段由特定的 Winsock 实施方案设定，事实上没有用。不要使用下面这两个字段：iMaxSockets 和 iMaxUdpDg，它们是假定的同时最多可打开多少套接字和数据报的最大长度。然而，要知道数据报的最大长度应该通过 WSAEnumProtocols 来查询协议信息。同时最多打开套接字的数目不是固定的，很大程度上和可用物理内存的多少有关。最后，lpVendorInfo 字段是为 Winsock 实施方案有关的指定厂商信息预留的。任何一个 Win32 平台上都没有使用这个字段。

对 WSAData 结构的参数介绍如下：

- wVersion：指定 Windows Sockets DLL 期待调用方使用的 Windows Sockets 规范的版本。
- wHighVersion：指定 DLL 可支持的 Windows Sockets 规范的最高版本。通常它和 wVersion 相同。
- szDescription：指向一个 NULL 结尾的 ASCII 字串，Windows Sockets DLL 将 Windows Sockets 实现的说明及厂商描述拷至该串。这段文本（长度最多 256 个字符）可能包含任何字符，但厂家注意到不要把控制和格式字符包含进去。该字串的最经常的应用就是在状态消息中显示。
- szSystemStatus：指定一个 NULL 结尾的 ASCII 字串，Windows Sockets DLL 将相关的状态和配置信息拷至该串。Windows Sockets DLL 只有在该信息对用户或支撑人员有用时才会使用该域，它不应该被认为是 szDescription 域的扩展。

- iMaxSockets：指定一个进程可以打开的最大套接口数目。Windows Sockets 的实现可以提供一个全局的套接口池给任何进程分配，也可以为每个进程分配套接口资源。该数字可反映出 Windows Sockets DLL 或网络软件是如何配置的。应用程序员可以使用该数字作为该 Windows Sockets 实现是否可以被应用程序使用的原始依据。例如，一个 X Windows 服务器可能在它启动时检查 iMaxSockets，若它小于 8，应用程序显示一条错误信息，让用户重新配置网络软件。（这是 szSystemStatus 可能使用到的一种情况）显然，这并不保证一个特定的应用程序可以实际分配到 iMaxSockets 个套接口，因为可能有其他的 Windows Sockets 应用程序在使用。
- iMaxUdpDg：指定以字节表示的可由 Windows Sockets 应用程序发送或接收的最大UDP数据报的大小。如果应用程序没有给出限制，iMaxUdpDg 为 0。在 Berkeley 套接口的许多实现中，对于 UDP 数据报的导向有一个隐含的限制，即 8192 字节。Windows Sockets 的实现可以在分配碎片重组缓冲区的基础上给出界限。对于一般的 Windows Sockets 实现 iMaxUdpDg 的最小值为 512。注意不考虑 iMaxUdpDg 的值，而试图在网络上发送一个大于最大传输单元（MTU）的广播数据报是不明智的（Windows Sockets API 没有提供发现 MTU 的机制，但它必须不小于 512 字节）。
- lpVendorInfo：指定指向厂商规定数据结构的远指针。该结构的定义（如果提供）超出了本规范的范围。

应用程序或 DLL 若需要多次得到 WSAData 结构信息，就必须多次调用 WSAStartup()。然而，wVersionRequested 参数假设在所有调用 WSAStartup()中都相同，也就是，应用程序或 DLL 不能在第一次调用 WSAStartup()后改变 Windows Sockets 的版本号。

对应于每一次 WSAStartup()调用，必须有一个 WSACleanup()调用，以使第三方（third-party）DLL 可以利用和应用程序相关的 Windows Sockets DLL。这意味着，如果应用程序调用了 WSAStartup()三次，它就必须调用 WSACleanup()三次。对 WSACleanup()的前两次调用除了减少内置计数器外不做任何事，对任务的最后一次 WSACleanup()调用，为任务释放了所有所需的资源。

WSAStartup 函数返回值如下。

（1）如果返回值为 0，则表示函数调用成功。

（2）否则返回下列的错误代码之一。注意通常依靠应用程序调用 WSAGetLastError()机制获得的错误代码是不能使用的，因为 Windows Sockets DLL 可能没有建立“上一错误”信息储存的客户数据区域。错误代码如下：

- WSASYSNOTREADY
  指出网络通信依赖的网络子系统还没有准备好。
- WSAVERNOTSUPPORTED
  指出所需的 Windows Sockets API 的版本未由特定的 Windows Sockets 实现提供。
- WSAEINVAL
  应用程序指出的 Windows Sockets 版本不被该 DLL 支持。

HOSTENT 结构以及 GetHostByName 函数的用法

对于用户来说，IP 地址是不容易记住的。因此，许多人更愿意用一个易记的、友好的主机名而不是 IP 地址。Winsock 提供了两个支持函数，它们有助于用户把一个主机名解析成 IP 地址。

Windows 套接字 GetHostByName 和 WSAAsynGetHostByName API 函数从主机数据库中取回与指定的主机名对应的主机信息。两个函数均返回一个 HOSTENT 结构，该结构的格式如下：

```
struct HOSTENT
{
    char FAR *          h_name;
    char FAR * FAR *    h_aliases;
    short               h_addrtype;
    short               h_length;
    char FAR * FAR *    h_addr_list;
};
```

Windows 中提供的 PHOSTENT 结构就相当于 hostent *，在此不做介绍。

对 HOSTENT 结构中参数的介绍如下：

- h_name：正式的主机名。如果网络采用了“域内命名系统（DNS）”，它就会导致命名服务器返回响应的“全限定域名（FQDN）”。如果网络使用一个本地“多主机”文件，主机名就是 IP 地址之后的第一个条目。
- h_aliases：一个由主机备用名组成的空中止数组。
- h_addrtype：表示即将返回的地址家族。
- h_length：对 h_addr_list 字段中的每一个地址定义字节长度进行定义。
- h_addr_list：一个由主机 IP 地址组成的空中止数组（可以为一个主机分配若干个 IP 地址）。这个数组中的每个地址都是按网络字节顺序返回的。一般情况下，应用程序都采用该数组中的第一个地址。但是，如果返回的地址不止一个，应用程序就会相应地选择一个最恰当的，而不是一直都用第一个地址。

GetHostByName API 函数的原型如下：

```
struct HOSTENT FAR * GetHostByName
(
  const char FAR * name
```

);

其中，name 参数表示准备查找的那个主机的名称。如果这个函数调用成功，系统就会返回一个指向 HOSTENT 结构的指针。

GetHostByName()返回对应于给定主机名的包含主机名字和地址信息的 HOSTENT 结构指针。返回的指针指向一个由 Windows Sockets 实现分配的结构。应用程序不应该试图修改这个结构或者释放它的任何部分。此外，每一线程仅有一份这个结构的拷贝，所以应用程序应该在发出其他 Windows Scokets API 调用前，把自己所需的信息拷贝下来。

GetHostByName()实现没有必要识别传送给它的 IP 地址串。对于这样的请求，应该把 IP 地址串当作一个未知主机名同样处理。如果应用程序有 IP 地址串需要处理，它应该使用 inet_addr()函数把地址串转换为 IP 地址，然后调用 GetHostByAddr()来得到 HOSTENT 结构。

GetHostByName 函数的返回值如下。

（1）如果没有错误发生，GetHostByName()返回如上所述的一个指向 HOSTENT 结构的指针，否则，返回一个空指针。应用程序可以通过 WSAGetLastError()来得到一个特定的错误代码。

（2）错误代码：

- WSANOTINITIALISED
  指出在应用这个 API 前，必须成功地调用 WSAStartup()。
- WSAENETDOWN
  指出 Windows Sockets 实现检测到了网络子系统的错误。
- WSAHOST_NOT_FOUND
  指出没有找到授权应答主机。
- WSATRY_AGAIN
  指出没有找到非授权主机，或者 SERVERFAIL。
- WSANO_RECOVERY
  指出无法恢复的错误，FORMERR，REFUSED，NOTIMP。
- WSANO_DATA
  指出有效的名字，但没有关于请求类型的数据记录。
- WSAEINPROGRESS
  指出一个阻塞的 Windows Sockets 操作正在进行。
- WSAEINTR
  指出阻塞调用被 WSACancelBlockingCall()取消了。

GetHostName API 函数

GetHostName API 函数的原型如下：

```
int GetHostName
```