Zero-day攻击多态蠕虫研究与进展

李高锋,何聚厚
(陕西师范大学 计算机科学学院,陕西 西安 710062)

摘 要:随着计算机网络的不断普及与发展,网络蠕虫已经成为网络系统安全的重要威胁之一。近年来,网络蠕虫又有了新的变化,出现了新的Zero-day攻击多态蠕虫,这种蠕虫采用"多态"技术并以"Zero-day漏洞"为攻击目标,可在短时间内有效地避开检测系统,成为未来互联网安全的一大隐患。因此,研究Zero-day攻击多态蠕虫及其检测技术是非常必要的。首先论述了Zero-day攻击多态蠕虫的攻击原理,接着对近几年提出的基于网络流过滤和模拟执行检测等方法进行了分析、总结,最后给出一些热点问题及展望。

关键词:网络安全;Zero-day攻击;多态蠕虫
中图分类号:TP301.6 **文献标识码**:A **文献编号**:1673-629X(2011)09-0216-05

Research and Development of Zero-day Polymorphic Worm

LI Gao-feng, HE Ju-hou
(School of Computer & Science, Shaanxi Normal University, Xi'an 710062, China)

Abstract: With the increasing popularity and development of computer network, internet worms have become one of the important network system security threats. In recent years, internet worms have a new change that is Zero-day polymorphic worms, which could use metamorphic techniques and regard Zero-day vulnerabilities as its attack targets, it can evade the detection of the existing IDSes in short time. Therefore, research Zero-day polymorphic worms and detection technique is very necessary. Zero-day polymorphic worm's attack mechanism is firstly presented, then some network-sifting and simulation- based detection techniques in recent years are analysed and concluded, and finally some hot questions are provided.

Key words: network security; Zero-day attack; polymorphic worm

0 引 言

1988年的Morris蠕虫[1]感染了6000多台邮件服务器,而这个数字占当时邮件服务器总数的10%,造成了高达9600万美元的经济损失。2001年的Code Red蠕虫[2]在爆发不到一天的时间内就攻击了25万台主机,造成了约20亿美元的重大损失。2003年8月的"冲击波"蠕虫[3]爆发8天内就造成了20亿美元的损失。2006年的"熊猫烧香"蠕虫[4]在短时间内感染了国内近千万台计算机,造成的损失难以估计。

传统的入侵检测系统(IDS)是基于特征码进行识别的(像Bro[5]和Snort[6])。近年来,"Zero-day漏洞"攻击迅速增多,再加上攻击者使用多态(Polymorphic)、变形(Metamorphic)等代码混淆技术,对基于特征匹配为主的安全技术提出了新的挑战。传统的入侵检测方法一般是对已知漏洞的攻击检测具有很高的效率,而新的"Zero-day漏洞"攻击是利用最新漏洞在很短时间内生成的攻击代码,这样一来,基于特征提取的入侵检测方法已经很难有效地进行检测了。

因此,对Zero-day攻击多态蠕虫的检测模型研究已成为目前网络安全领域的一个研究热点。

收稿日期:2011-02-21;**修回日期**:2011-05-26
基金项目:国家自然科学基金(60503008)
作者简介:李高锋(1982-),男,硕士研究生,研究方向为计算机网络安全;何聚厚,副教授,硕士生导师,研究方向为计算机网络安全。

1 Zero-day攻击多态蠕虫

1.1 Zero-day漏洞

Zero-day漏洞是指被攻击者掌握,但软件开发厂商还没发现或者未被公开的漏洞,对于这类漏洞还没有相应的补丁程序。通常,从新漏洞的发现到利用,不会超过24小时,由此就形像地把这种攻击称为Zero-day攻击,将被利用的新漏洞称为Zero-day漏洞[7]。公布漏洞的权威网站有如下两个:

(1)http://cve.mitre.org/ 该网站会对公布的漏洞进行编号,这种编号通常也是引用漏洞的标准方式。

(2)http://www.cert.org/ 该网站会在第一时间发布当前的严重漏洞、厂商的升级补丁链接和用户应该采取的临时措施等。

Zero-day漏洞是危害最大的漏洞,当然对于攻击者来说也是最有价值的漏洞。

1.2 Zero-day攻击原理

所谓Zero-day攻击,就是黑客会自己编写Zero-day漏洞的利用脚本,对系统或应用程序发动攻击[8]。

黑客实施一次Zero-day攻击会按如下的流程来进行:

(1)黑客发现漏洞;

(2)编写漏洞利用脚本;

(3)寻找目标发动攻击;

(4)进入目标网络或系统;

(5)安装后门控制系统或得到网络中机密数据;

(6)清除攻击痕迹。

1.3 Zero-day攻击多态蠕虫

Zero-day攻击多态蠕虫是一类检测难度大、利用多种技术来规避现有检测技术的蠕虫。

Zero-day攻击多态蠕虫通过加密、多态、变形、模拟等代码混淆技术,来改变自身的存在形式和表现状态,使其自身在执行时在字节序列上看起来互不相同,但是其功能是不变的。在传播过程中,可利用缓冲区溢出攻击、二进制代码注入等攻击方式来完成攻击,其在传播过程中,有一些不变量(如漏洞触发的条件和返回地址等),多态蠕虫体(见图1)通常和一般蠕虫攻击一样,包含有获得主机权的代码和传播代码,其不同的地方是,代码体中包含有会产生不同变体的多态引擎代码,其通常使用不同的变形引擎技术来达到变形目的[9]。

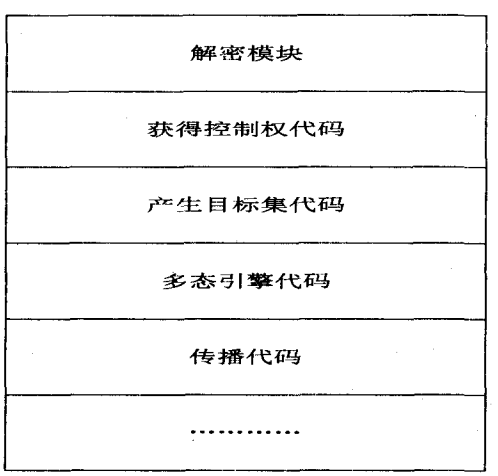

图1　多态蠕虫体结构

1.4 多态和变形技术

1.4.1 多态技术(Polymorphic)

蠕虫变形技术最初采用的方法是,通过加密技术来改变蠕虫代码,加密技术采用的方法比较简单。为了使具体的单个蠕虫代码实例的二进制形式都不相同,其密钥都会发生改变,但是解密代码是保持不变的。因此,对于蠕虫检测系统来说,解密代码可作为检测的特征码。

使用加密技术编写的蠕虫很容易就能被检测出来,为了克服加密技术的不足,蠕虫编写者在加密技术的基础上引入了多态技术,这样使得解密代码也表现出多样性。

图2给出了多态技术的一般工作流程[10]:

(1)初始解密代码:对原始的解密代码做合法性检查,如果没有错误进入下一个环节。

(2)指令顺序变换:通过改变部分指令的顺序来达到改变解密代码的表现形式,而这些指令相对顺序改变不会影响解密代码的执行。

(3)寄存器重命名:对寄存器进行重命名,并且不同的实例使用不同的寄存器组。

(4)指令替换:包含三种类型:①指令扩展变换,指将原来一条指令用等价的几条新指令替换;②指令收缩变换,指将原来多条指令用等价的一条新指令替换;③ 等价指令替换,将原来一条指令替换成等效的一条新指令。

(5)指令插入:在解密代码中,插入两种类型的指令:①无用指令插入,插入push/pop eax等nop指令;②垃圾指令插入,插入实际执行效果的指令。

(6)新的解密代码:对原有解密代码,经过一系列的变换、重命名、替换、插入等形成新的解密代码。

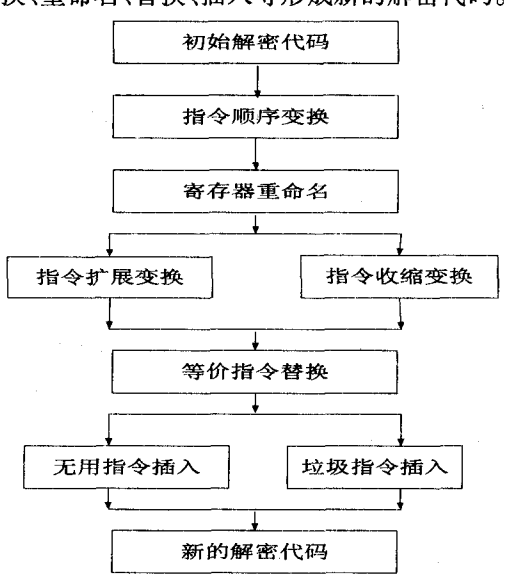

图2　多态技术的工作流程

1.4.2 变形技术(Metamorphic)

为了增强多态蠕虫的生存能力,编写者通常会使用像ADMmutate、PHATBOT、CLET和JempiScodes等多态变形引擎来书写蠕虫代码。

变形技术首先对蠕虫代码进行反汇编,得到"内部伪码"。为了避免产生过大的程序体,再对其进行

压缩。接着使用相应的多态技术对压缩代码进行变换,最后进行汇编生成可执行的新代码体。变形技术对解密代码和恶意攻击代码都进行了变换,使得其达到两个目的:①保持蠕虫的攻击行为不变;②保持不同蠕虫实例(变种)的具体内容完全不同。

1.4.3 典型的 Shellcode 多态技术

蠕虫的多态主要来自 shellcode 的多态。多态 shellcode 的典型结构如图3所示[11]。

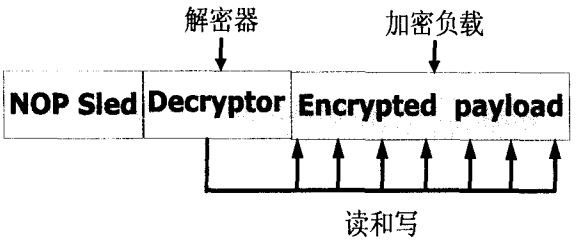

图3 多态 shellcode 的结构

首先,使用单字节或多字节指令 NOP Sled,接着执行解密器(Decryptor)部分,解密器的作用是还原出 shellcode 的蠕虫代码、多态引擎等,多态引擎产生蠕虫病毒实例,最后,在内存中读取负载内容,程序流程就转入解密出的原始 shellcode,开始执行真正的 shellcode 功能。

2 检测技术

由于 Zero-day 攻击多态蠕虫在攻击过程中,利用了 Zero-day 漏洞,使用了多态、变形等技术,而蠕虫一旦发作,在很短的时间里就会造成巨大的危害,因此,需要提出一种自动的多态蠕虫检测方法。近几年来,针对 Zero-day 攻击多态蠕虫,许多研究者提出了许多自动检测模型和算法,总结起来主要有两大类:一种是基于网络流过滤的自动检测方法;一种是基于模拟执行的自动检测方法。

2.1 基于网络流过滤的自动检测

Zero-day 攻击多态蠕虫在传播过程中有其不变的部分,即:数据包内容的相似性。这种检测方法是从可疑网络流中自动获取蠕虫的特征(不变部分),并利用这些特征完成对 Zero-day 攻击多态蠕虫的检测。

2004年,H. A. Kim 等人提出 Autograph[12]。其基本思路是:首先对正常流量和可疑流量进行粗略的分类,再对可疑流量做字符串划分,它采用的是非定长的字符串划分方法,使用了 Rabin Fingerprint[13]来提取特征码。其基本特点是:

①Autograph 将正常流量和可疑流量分别采用比较简单的分类器,提高了其与不同分类器之间的兼容性;

②Autograph 还采用了白名单方式,提高了特征码的准确性;

③Autograph 提取出的特征码,将以 Bro 的特征格式输出;

④Autograph 只是针对于 TCP 方式的蠕虫攻击。

2004年,S. Singh 等人提出 EarlyBird[14]。其基本思路是:通过一段公共的恶意代码序列和特定的数据源地址被分散的程度这样两个关键的行为特征来快速检测未知的蠕虫攻击。

而 EarlyBird 的检测方法与 Autograph 不同:

①EarlyBird 没有预处理过程,它对每一个数据包都进行处理,不仅包括 TCP 流,还包括 UDP 流;

②EarlyBird 对 TCP 流没有进行流重组,而只是单独处理每一个 IP 数据包,比 Autograph 节省时间,但难以发现以分片方式进行传播的蠕虫;

③EarlyBird 采用固定长度的滑动窗口来划分序列(窗口大小为39字节);

④EarlyBird 提取出的特征码,将以 Snort 的特征格式输出。

Autograph 和 EarlyBird 都是检测蠕虫的早期方法,它们生成的特征码都是"单一连续的子节串",可以检测到变种的新型蠕虫攻击特征,但却不能有效提取多态蠕虫的特征。

2004年,K. Wang 等人提出 PAYL[15]。它是基于数据包流量的异常检测,不仅针对 Zero-day 蠕虫,还能检查更大范围的攻击。该系统比较进入和流出系统的数据包,检查二者所包含的子串或子序列是否相等,比较出的结果作为特征码提取出来,但其误报率较高。

2005年,J. Newsome 等人提出 Polygraph[16]。它能够提取多态蠕虫的特征码,Polygraph 能提取多态蠕虫的三类特征码:①有序子串集合;②无序子串集合;③带 Bayes 权值的子串集合。其检测思路是:通过一个连续的字节序列令牌("token")来提取特征码,Polygraph 通过将从捕获的样本中提取出至少出现 k 次且长度不小于 a 的令牌,而且一个令牌不是另一个的子串,这些不同的令牌组合成了多态蠕虫的特征码。在特征码提取之前,Polygraph 需要一个流量分类器根据端口号的不同来重组各个数据包流量,但是作者并没有明确所使用的分类器,而且令牌在提取过程中可能会丢失重要的短序列。

2006年,Z. Li 等人提出 Hamsa[17]。它是针对 Zero-day 攻击多态蠕虫,它假设所有攻击都会包含一些不变内容(即固定部分),因而蠕虫的不同变种肯定包含一些固定内容。Hamsa 首先会对可疑网络流进行标记,其次在可疑数据流中提取出子串,Hamsa 认为可能会出现误报,因为可疑数据流中很可能有一部分正常数据流。所以,能作为蠕虫特征的子串应该满足这样两个前提:①在可疑数据流中,子串是经常出现的;②

在正常数据流中,子串是很少出现的。Hamsa只针对固定内容,因此,它很难检测出使用了多种变形和加密技术的蠕虫。

2008年,Mohssen等人提出Honeycyber[18]。它可以实现自动快速地检测零日攻击多态蠕虫,该系统设计了一个新的双蜜网系统,能够自发检测新蠕虫的攻击和及时对可疑流量进行隔离,该系统具有较低的误报率和漏报率。

2.2 基于模拟执行的自动检测

单一地使用网络流过滤和特征码提取技术,对部分多态蠕虫的检测效果是不佳的,而多态蠕虫的攻击行为特征码是不会发生变化的。基于此,许多研究者提出利用模拟蠕虫执行的方法,进而获取蠕虫的行为特征,最后达到自动检测Zero-day攻击多态蠕虫的目的。

2005年,Newsome等人提出Taintcheck检测系统[19]。该系统采用了Dynamic Taint Analysis思想,在模拟环境下模拟执行可疑代码,利用动态追踪,把网络的输入流与其控制流的改变关联起来,根据这些关联来查找恶意输入进而推断出蠕虫的属性。作者采用的模拟器是valgrind。检测分为三步:①将来自于不可信源的数据标记为taint。②跟踪标记数据的扩散。③报告标记数据的非法使用。最终使用导致攻击爆发的关键数据作为特征,作者将来自于socket的网络数据包标记为taint。这种检测方式能检测大多数恶意代码,并且效率高,误报、漏报问题极少。也能产生特征,不过特征过于简单。

2005年,J. R. Crandall等人提出DACODA(DAvis malCODe Analyzer)[20]。它是通过多态蠕虫的代码来推导Zero-day漏洞,其使用Minos作为Oracle来捕获攻击。DACODA可以针对"数据在主机处理过程中不同阶段的情况"和"数据在网络流中的情况"。与TaintCheck一样,DACODA利用动态追踪,把网络的输入流和数据控制流的改变关联起来,以此来查找恶意输入,进而推断蠕虫的属性。DACODA可用于Honeypot技术中。

2006年,Portokalidis等人提出argos检测系统[21],作者同样采用Dynamic Taint Analysis思想,在模拟环境下模拟执行可疑代码。采用的模拟器是qemu。不同于Taintcheck检测方法的是作者通过插入一段shellcode提取被攻击者信息,包括进程id、端口号。结合导致攻击爆发的关键数据和被攻击者信息产生特征,最后将特征传送给Sweetbait进行优化。这种检测方式具有Taintcheck的所有优点,并且能够产生更加精确的特征。

2007年,Costa等人提出Bouncer检测系统[22],作者通过标记一个攻击样本的数据、跟踪标记数据的状态、收集其执行路径,产生一个初始的过滤器。也就是说,将样本执行过程中所有的条件语句以标记的形式添加到过滤器中。但是作者考虑到攻击者可以通过改变执行路径的方式躲避检测,因此提出3步优化方案:①消除过滤器中不需要的条件。②使用一个简单的条件集合表示每个常用的库函数。③在前两步优化的基础之上,使用样本中的爆发信息产生新的能够导致相同程序漏洞攻击的实例。最后结合每个生成的实例的过滤条件产生一个最终的高效的过滤器,依此作为攻击检测的依据。

2008年,Portokalidis等人又提出Eudamon检测系统[23]。同argos不同的是Eudamon是基于主机而非网络,为了提供个人机的实时防护并且不影响个人机的性能,作者提出了按需求使用的方式,即在用户不操作主机的情况下进行防护。同样采用Dynamic Taint Analysis思想,在模拟环境下模拟执行可疑代码。不过作者加入了部分功能实现程序执行流程转移到模拟环境和从模拟环境恢复到真实环境。这种检测方法不同于上述几种,针对的是个人主机,扩展了网络安全研究的新领域。

3 结束语

尽管已经对入侵检测技术有了大量的研究,但目前的网络安全形势依然严峻,对于Zero-day攻击多态蠕虫,其研究热点有:

(1)"Zero-day漏洞"利用技术研究:为了能够有效地检测Zero-day攻击,首先要尽早发现未知的Zero-day漏洞,其次要深入研究可能的利用"Zero-day漏洞"进行攻击的原理。因此,"Zero-day漏洞"发现及利用技术有待进一步研究。

(2)Zero-day攻击多态蠕虫的仿真模拟:Zero-day攻击多态蠕虫的危害大、检测难,可以根据多态蠕虫所使用的传播策略及相关算法,使用仿真软件构建一个安全的模拟环境来执行真正的蠕虫,这样就能很方便地观察蠕虫的传播、感染和攻击的全过程。这为设计检测和预防系统提供了可靠的数据和环境。因此,Zero-day攻击多态蠕虫的仿真模拟是今后研究的一个重要方向。

(3)Zero-day攻击多态蠕虫检测技术:越来越多的蠕虫引入了多态和变形、加密和代码混淆、动态伪装模拟正常流量、攻击后自动消除痕迹、阻止反汇编等技术,来阻止对其检测。对于Zero-day攻击多态蠕虫的复杂性,单纯地使用一种方法已经很难有效地对其进行检测,因此,如何将多种检测方法高效地结合起来,是今后研究与发展的一个主要方向。

参考文献：

[1] Spafford E H. The Internet Worm Program: An analysis[R]. West Lafayette: Department of Computer Science, Purdue University, 1988:1-29.

[2] EEye Digital Security. Code Red worm[EB/OL]. 2001. http://www.eeye.com/html/research/advisories/a120010717.html.

[3] 潘春华,刘寿强,张 健,等. 冲击波蠕虫病毒的解析与防治[J]. 计算机安全,2003(9):68-72.

[4] 陶 金. 从"熊猫烧香"看新型网络病毒的特点和防御方法[J]. 辽宁师专学报(自然科学版),2007,9(1):50-51.

[5] Paxson V. Bro:a system for detecting network intruders in real - time[C]//Proceedings of the 7th USENIX Security Symposium. San Antonio, TX: USENIX Association, 1998:31 - 51.

[6] 陈 伟, 周继军,许德武. Snort 轻量级入侵检测系统全攻略[M]. 北京:北京邮电大学出版社 ,2009.

[7] 王 清. 0day 安全:软件漏洞分析技术[M]. 北京:电子工业出版社,2008.

[8] 李 贷. 基于零日漏洞攻击的原理与防范[J]. 电脑知识与技术,2009(33):9394-9395.

[9] 徐晓萌,郭山清,徐秋亮. 多态蠕虫的研究与进展[J]. 计算机科学与探索,2008(2):131-138.

[10] 张岗岭,何聚厚. 基于模拟执行的恶意行为检测模型研究[D]. 西安:陕西师范大学,2010.

[11] 王兰佳. 基于网络的 0-day 多态蠕虫检测算法研究[D]. 北京:清华大学,2009.

[12] Kim H A, Karp B. Autograph: toward automated, distributed worm signature detection[C]//Proceedings of the 13th USENIX Security Symposium. [s. l.]:[s. n.],2004.

[13] Rabin M O. Fingerprinting by Random Polynomials[R]. Center for Research in Computing Technology, Harvard University, 1981.

[14] Singh S, Estan C, Varghese G, et al. Automated worm fingerprinting[C]//Proceedings of the 6th ACM/USENIX Symposium on Operating System Design and Implementation (OSDI). [s. l.]:[s. n.],2004.

[15] Wang K, Stolfo S. Anomalous payload-based network intrusion detection[C]//RAID. [s. l.]:[s. n.], 2004.

[16] Newsome J, Karp B, Song D. Polygraph: Automatically generating signatures for polymorphic worms[C]//In:IEEE Security and Privacy Symposium. [s. l]:[s. n.],2005.

[17] Li Z, Sanghi M, Chen Y, et al. Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience[C]//In IEEE Symposium on Security and Privacy. Oakland, CA:[s. n.],2006.

[18] Mohammed M M Z E, Chan H A, Ventura N. Honeycyber: Automated signature generation for zero - day polymorphic worms[C]//IEEE Conferences. [s. l.]:[s. n.],2008:1-6.

[19] Newsome J, Song D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software[C]//Proceedings of the 12th Annual Network and Distributed System Security Symposium. San Diego, USA:[s. n.],2005:221-237.

[20] Crandall J R, Su Z, Wu S F. On deriving unknown vulnerabilities from zero - day polymorphic and metamorphic worm exploits[C]//Proc. of ACM CCS. [s. l]:[s. n.], 2005.

[21] Portokalidis G, Slowinska A, Bos H. Argos: an emulator for fingerprinting zero - day attacks[C]//Proc. of the 1st ACM SIGOPS EUROSYS. Leuven, Belgium:[s. n.],2006.

[22] Costa M, Castro M, Zhou L, et al. Bouncer: securing software by blocking bad input[C]//SOSP. [s. l.]:[s. n.],2007:117-130.

[23] Portokalidis G, Bos H. Eudaemon: Involuntary and On-Demend Emulation Against Zero-Day Exploits[C]//Proc. of the 3rd ACM SIGOPS EUROSYS. [s. l.]:[s. n.],2008:287-289.

(上接第 110 页)

统镜像的移植。

参考文献：

[1] 于 明,盛 俊,刘建设. 基于 Intel PXA270 的系统引导程序移植与实现[J]. 计算机技术与发展,2010,20(6):32-35.

[2] Intel PXA270 Processor Family Developer's Manual[M]. USA:Intel,2004.

[3] 陈章龙. 嵌入式技术与系统-Intel Xscale 结构与开发[M]. 北京:北京航空航天大学出版社,2004.

[4] 黄智伟,邓月明,王 彦. ARM9 嵌入式系统设计基础教程[M]. 北京:北京航空航天大学出版社,2008.

[5] 吴 涛,张 翼,杨 鹏. 基于 Intel PXA27x 的系统引导程序(Blob)移植与实现[J]. 嵌入式软件应用,2007,23(8):69-71.

[6] Suthar S. U-boot Porting guide[J]. Einforchips Technology, 2007(4):6-7.

[7] 白伟平,包启亮. 基于 ARM 的嵌入式 bootloader 浅析[J]. 微计算机信息,2006,22(4):99-100.

[8] 王文东,李竹林,尚建人. 汇编语言与 C 语言的混合程序设计技术[J]. 计算机技术与发展,2006,16(8):19-20.

[9] 徐亚鹃,谢凯年. 用 U-Boot 构建 IXP2350 目标系统的引导程序[J]. 计算机技术与发展,2007,17(5):10-14.

[10] 刘 淼. 嵌入式系统接口设计与 Linux 驱动程序开发[M]. 北京:北京航空航天大学出版社,2006.

[11] 兰少华,杨余旺,吕建勇. TCP/IP 网络与协议[M]. 北京:清华大学出版社,2006.

[12] 罗军舟,黎波涛,杨 明. TCP/IP 协议及网络编程技术[M]. 北京:清华大学出版社,2004.