基于 LIBPCAP 的网络流量实时采集与信息萃取*

胡文静[1,2]，李 明[1]，刘锦高[1]

(1. 华东师范大学 信息科学技术学院，上海 200062；2. 湖南理工学院 物理与电子信息系，湖南 岳阳 414006)

摘 要：论述了网络流量实时采集与信息萃取的基本原理与实现方法。重点阐述在 Linux 环境下基于 LIBPCAP 包捕获机制，并在 QT3.3 平台上成功实现了基于 GUI 界面的网络流量实时采集与信息萃取及 C++ 类的封装。

关键词：网络流量；TCP/IP；LIBPCAP；实时数据采集；包捕获；信息萃取；GUI；DDoS 攻击检测

中图法分类号：TP393.08 **文献标识码**：A **文章编号**：1001-3695(2006)06-0236-03

Real-time Traffic Capturing and Information Extracting of Raw Traffic Data Based on LIBPCAP

HU Wen-jing[1,2], LI Ming[1], LIU Jin-gao[1]

(1. *College of Information Science & Technology, East China Normal University, Shanghai* 200062, *China*; 2. *Dept. of Physics & Electronics Information, Hunan School of Science & Technology, Yueyang Hunan* 414006, *China*)

Abstract: This paper gives the principle and the realization of real-time traffic capturing and information extracting of raw traffic data. We emphasize on the mechanism of packet capturing based on LIBPCAP under Linux system, the realization based on GUI interface and the C++ class encapsulation as well.

Key words: Network Traffic; TCP/IP; LIBPCAP; Real-time Traffic Capturing; Packet Capturing; Information Extracting; GUI; DDoS Attacks Detection

社会各部门和机构(如银行或政府部门)越依赖于计算机网络(如互联网)，计算机网络安全就越重要。如果一个银行网站因为受攻击而拒绝服务合法客户，那么就意味着该银行的名誉受损和客户丢失，从而导致资金流失。如果一个政府网站因为受攻击而拒绝服务合法客户，那么政府的名誉受损。2000 年 2 月网络黑客对全球著名电子商务网站(Yahoo, eBay.com, eTrade.com 等)成功实施分布式拒绝服务攻击(DDoS)后，检测 DDoS 攻击就成为极其迫切的重要研究课题[5]。在入侵检测方面有两类系统，即基于主机(Host-based)的入侵检测系统(IDS)和基于网络(Network-based)的 IDS[6]。基于网络的 IDS 以网络流量为检测对象[6]，它又可分为两类：误用检测(Misuse detection)和异常检测(Anomaly detection)[7]。我们在文献[4]提出的系统属异常检测。所以，网络流量的实时采集是基于网络的 IDS 的第一个环节，也是关键环节。

网络流量是数据包流(Packet stream)。一个包有若干属性(如 IP 地址、包尺寸、协议类型等)。理论上，包的每个属性均可以用于检测。因此，作为基于网络的采用异常检测策略的 IDS 的第一个环节，我们不但要求采集系统能实时记录数据包流，而且要根据特定的检测系统提供相应信息(即信息萃取)。例如，文献[4]所述系统需要的是包尺寸时间序列，而文献[8]所述的系统需要包头(Packet head)的数据信息。针对异常检测的这个特点，我们设计了一个网络流量的实时采集与信息萃取系统。这个系统不仅是我们在文献[4]中介绍的以包尺寸时间序列为检测对象的 IDS 的数据采集环节，也可以方便地嵌入到用其他包属性(如包头)组成的异常检测系统。

传统的网络流量数据采集系统(又叫做网络流量嗅探器)有 Tcpdump[10]，Tethereal (Linspire Inc.)[10]，Tcptrace (Shawn Ostermann)[11]等。这些采集系统为网络管理员掌握网络运行状况带来了诸多便利。但我们对这些软件包进行了反复使用分析后认识到这些软件包有以下共同特点：①它们都是基于控制台命令行参数方式运行。人机界面适合于软件专业人员，对一般的工程师或网络安全人员或管理员欠友好，并需要用户熟悉其复杂的运行参数。②各软件包均自成一体，代码的复用率低，难以嵌入到各种特定的入侵检测系统。③检测输出信息以控制台输出为主或以重定向方式输出至文本文件，从中获取针对性强的有用信息对一般研究人员和网络管理员不方便甚至相当困难。鉴于以上所述，在设计 IDS 时采用了基于 GUI 界面的可重用设计方法，将网络流量实时采集与信息萃取模块进行了类的封装，使得采集系统具有以下优点和特点：

(1) 基于 GUI 界面风格，人机界面友好；

(2) 实时采集与信息萃取模块封装成标准的 C++ 类，代码复用率高；

(3) 可嵌入到其他入侵检测系统，作为其前端流量采集与信息萃取模块；

(4) 源代码开放，用户可在遵守 GPL 规范的基础上进行修改以适应其特定需求；

(5) 流量解析结果以 GUI 界面输出，且可以多种文件格式进行重定向，以满足其他如 MATLAB, Mathmatica 等统计分析软件的要求，有针对性地为研究人员提供可用信息。同时可以

收稿日期：2005-03-23；修返日期：2005-06-25

基金项目：上海市科委基金资助项目(047252001)

向各种基于网络的入侵检测系统提供相应的流量数据信息。

(6) 网络流量可以不同统计图形方式(如二维流量趋势、某段时间内的流量自相关函数等) 显示, 形象直观。

1　网络数据包捕获机制

1.1　网络数据包分类

TCP/IP 协议族中网络数据包按照其包含信息内容的不同可以分为以下几类[12,13]:

(1) 网络节点端口数据包

此类数据包指的是数据链路层(Link Layer) 网络节点设备端口流入和流出的数据包, 其内容包括包大小(Packet Header Length)、包捕获时间(Time-Stamp)、包类型(如 IP, ARP, RARP, PPPoE 等)、节点设备数据包的源地址(Ethernet Source Address) 与目的地址(Ethernet Destination Address)、数据包的个数(Total Packets)、丢包数(Dropped Packets) 等信息。

(2) IP 数据包

IP 数据包是指在网络层(Network Layer) 从源 IP 地址到目的 IP 地址的数据包, 比网络节点端口数据包包含了更为丰富的信息, 如源 IP 地址、目的 IP 地址、数据包传输类型(如 TCP, UDP, ICMP 等)、使用的 IP 版本(IPv4 或 IPv6) 等其他标准 IP 协议信息, 是网络分析、规划、设计和优化的重要依据。

(3) 传输层数据包

传输层数据包除了包含端到端 IP 数据包的信息之外, 还包含了传输层(TCP/UDP) 信息, 如 TCP 包的源端口与目的端口、TCP 包数据长度、TCP 连接请求/应答标志信息、UDP 端口、UDP 数据长度及数据纠错信息。

(4) 应用层数据包

应用层数据包包含了特定应用的详细信息, 如远程登录的 Telnet、文件传输协议(FTP)、用于发送电子邮件的 SMTP(Simple Mail Transfer Protocol)、SNMP(Simple Network Management Protocol), 对于安全、性能等方面的分析非常有效。

1.2　LIBPCAP 简介

LIBPCAP 实质上是一个与系统独立的 API 接口函数库, 由洛仑兹伯克利国家实验室开发, 用于用户层次的数据包截获工作。LIBPCAP 统一了不同系统中用于数据包截获的不同类型接口, 并使得类似的高层应用程序的编写和移植工作变得简单有效, LIBPCAP 库还在不断地进行更新和升级。

LIBPCAP 库所提供的主要功能函数有(参阅 LIBPCAP 手册[9]): ①pcap_open_live() 为获取捕获数据包的描述府, 用于查看网络数据包的传输; ②pcap_lookupdev() 为返回供 pcap_open_live() 使用的设备指针; ③pcap_open_offline() 为打开数据包文件以供离线分析; ④pcap_dump_open() 为打开文件以供写入数据包; ⑤pcap_setfilter() 为设置数据包过滤器程序; ⑥pcap_loop() 为启动数据包捕获。

1.3　数据包捕获

1.3.1　数据包捕获流程图

通过调用 LIBPCAP 库函数实现网络数据包的捕获, 程序流程如图 1 所示。

1.3.2　数据包捕获线程类

(1) 定义数据包捕获线程类成员变量

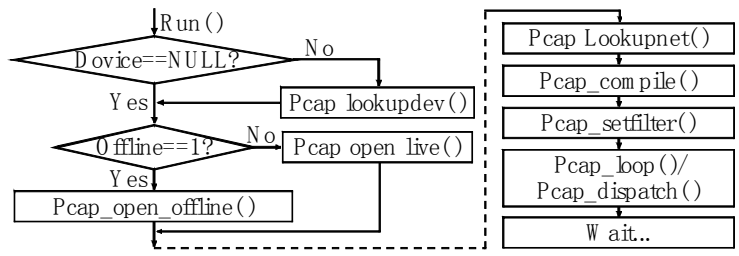

图 1　数据包捕获流程图

```
pcap_t *  pcap_fd;
QString WriteFilename;
QString ReadFilename;
QString WriteFile;
QString filters;
QString Device;
```

(2) 数据包捕获初始化成员函数

```
pcap_t*  CapThread: : pcap_init( char *  dev, char *  filter, int snap,
int timeout, int dumplevel)
{ //定义局部变量
    if (  dev = = NULL )
    if ( ( dev = pcap_lookupdev( errbuf ) ) = =  NULL)
      return NULL;
    if ( readfrom) {  //offline 捕获模式
      if ( (  _link = CheckFileHeader( ReadFilename) ) = = 0)  return
            NULL;
      else
        if ( ( p = pcap_open_offline( ReadFilename, errbuf) ) = = NULL)
              return NULL; }
        else //实时捕获模式
          if( ( p = pcap_open_live( dev, snaplen, 1, timeout, errbuf) ) = =
              NULL) return NULL;
        if ( pcap_lookupnet( dev, &ip, &mask, errbuf)  = =  - 1)  return
NULL;
        if ( pcap_compile( p, &bpf, filter, 1, mask)  < 0)  return NULL;
        if ( dumplevel >  = 0) {
          bpf_dump( &bpf, dumplevel ) ;
          return NULL; }
        else if ( pcap_setfilter( p, &bpf)  = = - 1)  return NULL;
    return p;
}
```

(3) 数据包捕获线程类成员函数

```
void CapThread: : pcap_cap( pcap_t*  p)
{ //构造回调函数用户数据结构(详见 2.1)
  if ( dumpto) {
    pcap_dumper_t
  * pp = pcap_dump_open( p, WriteFilename) ;
  pcap_userdata-> arg = pp; pcap_userdata- > p = this;
  pcap_loop( p, cnt, my_callback, ( u_char* ) pcap_userdata) ; }
  else{
    pcap_dumper_t
  * pp = pcap_dump_open( p, " tmp. dmp" ) ;
    pcap_userdata-> arg = pp; pcap_userdata-> p = this;
    pcap_loop( p, cnt, my_callback, ( u_char* ) pcap_userdata) ; }
    return;
  }
```

2　数据包信息萃取

2.1　包捕获回调函数用户数据结构*

在启动数据包捕获时, 必须将回调函数的地址传给 pcap_loop() 或 pcap_dispatch() 函数, 以便捕获到数据包后能调用回调函数(如 my_callback) 进行数据包的解析(图 2), 但由于回调函数的特殊性决定其不能作为普通成员函数进行封装, 这样回调函数与线程类之间的数据共享就成为线程类封装必须解决的关键问题。我们通过巧妙地改造回调函数用户参数(pcap_userdata), 将线程类对象指针及其他用户参数传入回调函数,

实现了回调函数和线程类之间的数据共享,具体实现如下:

```
//…
class CapThread; //线程类
struct CapPara{ //自定义数据结构
pcap_dumper_t* arg; //用于写数据包的文件指针
    CapThread* p; //线程类指针
};
//回调函数获取线程类指针的示例代码
void my_callback( u_char* args, const struct pcap_pkthdr* pkthdr, const u_
char* packet){
u_char* wf;
struct CapPara* pp = ( struct CapPara* ) args;
//获取用户自定义数据结构
CapThread* p = pp-> p; //获取线程类对象指针
wf = ( u_char* ) pp-> arg;
//获取除线程类对象指针外的其他用户参数
//…}
```

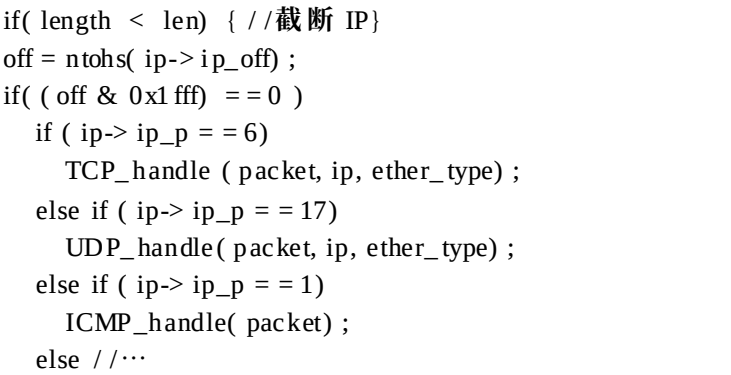

图 2 数据包解析流程图

2.2 数据包解析线程类成员函数

根据 TCP/IP 协议族层次结构图及各层数据包格式的定义[12, 13],将各层所加入的头和控制信息进行解封装,即可实现数据包解析。限于篇幅,这里只给出网络层数据包解析线程类成员函数:

```
u_char* CapThread:: handle_IP( struct pcap_pkthdr* pkthdr, u_char*
packet, u_int16_t ether_type)
{ //定义局部变量 ip, pppoe, length, hlen, off, version, len
    if ( ether_type == ETHERTYPE_IP) {
      ip = ( struct my_ip* ) ( packet + sizeof( struct ether_header) ) ;
      length -= sizeof( struct ether_header) ;}
    else if ( ether_type == ETHERTYPE_PPPOE) {
      pppoe = ( struct my_pppoe* ) ( packet + sizeof( struct ether_
              header) ) ;
      ip = ( struct my_ip* ) ( packet + sizeof( struct ether_header) +
           sizeof( struct my_pppoe) ) ;
      length -= ( sizeof( struct ether_header) + sizeof( struct my_
                pppoe) ) ;}
    else{
      ip = ( struct my_ip* ) ( packet + sizeof( struct ether_header) ) ;
      length -= sizeof( struct ether_header) ;}
    if( length < sizeof( struct my_ip) ) return NULL;
    len = ntohs( ip-> ip_len) ;
    hlen = IP_HL( ip) ; version = IP_V( ip) ;
    if( length < len) { //截断 IP}
    off = ntohs( ip-> ip_off) ;
    if( ( off & 0x1fff) == 0 )
      if ( ip-> ip_p == 6)
        TCP_handle ( packet, ip, ether_type) ;
      else if ( ip-> ip_p == 17)
        UDP_handle( packet, ip, ether_type) ;
      else if ( ip-> ip_p == 1)
        ICMP_handle( packet) ;
      else //…
    return NULL;
}
```

3 测试结果

我们在 Red Hat Linux 9 系统下对数据包捕获解析进行了测试,数据来源于 MIT 实验室(DARPA '99 Data),图 3 ~图 6 为 IP 数据包各层解析结果。

图 3 链路层解析结果

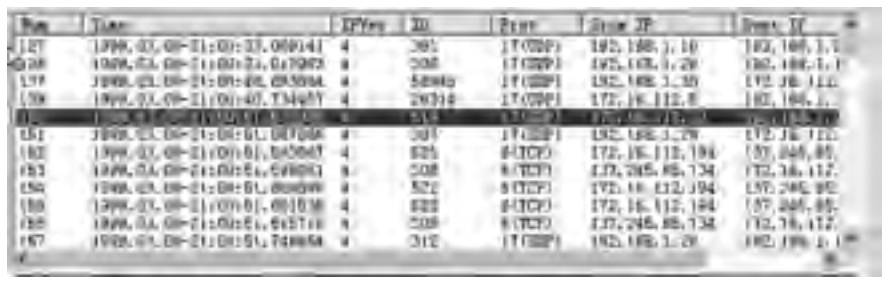

图 4 网络层解析结果

图 5 传输层解析结果

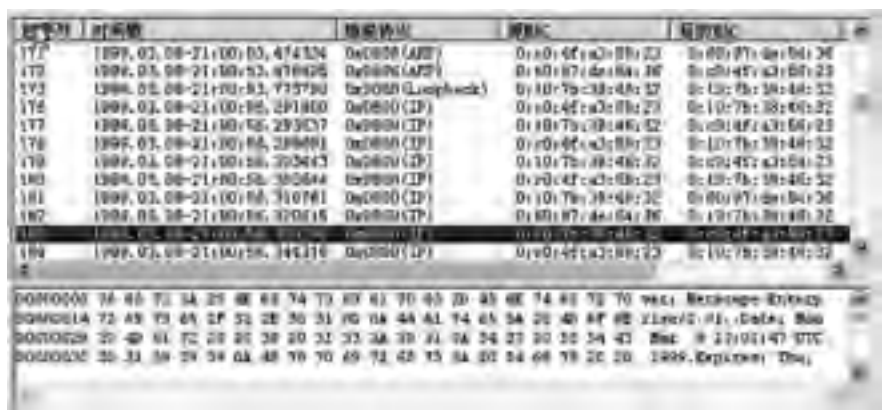

图 6 应用层解析结果

4 结束语

网络数据包实时采集与信息萃取是基于网络的入侵检测系统的第一个环节,是整个检测系统的一个关键。本文提出的数据采集与信息萃取系统比传统的数据采集系统(如 Tcpdump 或 Tcptrace)有四个优点:①实时性。能实时捕获网络流量,并交由后续分析模块进行信息萃取与分析。②可复用性。网络流量采集与信息萃取被封装成标准的C++类,提高了代码的可复用率,而且可方便地嵌入其他入侵检测系统,作为前端流量采集与信息萃取模块。③可定制性。由于此软件包源代码开放,用户可在遵守 GPL 规范的基础上对代码进行修改,以适应特定入侵检测系统的需要。④界面友好。采用 GUI 界面风格,人机接口友好;且信息能以多种文件格式输出,满足其他如 MATLAB, Mathmatica 等统计分析软件的需要。限于篇幅,文中只列出了数据包捕获与信息萃取几个关键的类成员函数。

参考文献:

[1] A Kemmerer, G Vigna. Intrusion Detection: A Brief History and Overview[J]. Supplement to IEEE Computer (IEEE Security & Privacy), 2002, 35(4):27-30.

[2] E Schultz. Intrusion Prevention[J]. Computer & Security, 2004, 23(4): 265-266.

(下转第 241 页)

分配给厨房…, 并将消息的格式修改如下:

设备类型 ID	该设备的物理空间标志码	操作码 ID

那么通过控制终端发送控制消息 Light Kitchen On 就可以打开厨房的灯, 这样无论家庭信息网络发生怎样的变化, 用户的控制消息内容是不变的, 仅仅改变的是数量, 即增加或减少某些相应的控制消息。

(2) 信息融合。智能家居环境网关与不同控制终端以及家庭信息网络进行通信, 收发不同协议的数据, 进行信息的融合。本模型中, 我们要实现对无线 PDA 信号的接发、手机短信的接发、网页表单信息的提取以及响应网页的发送。

①手持终端 PDA 与家庭信息网络的通信。无线通信模块采用 CC1000RF 芯片开发, 具有功耗低、传输距离适中的特点。无线通信模块通过串口连接到智能家居环境网关, 其数据的收发是通过对串口操作进行的。由于网关采用的是嵌入式 Linux 系统, Linux 把所有的硬件资源都虚拟成文件, 通过虚拟设备文件统一管理硬件设备, 要访问一个串口, 只需打开相应的设备文件, 然后向这个文件读写数据就可以完成数据的接收和发送。

②GSM 网络与家庭信息网络的通信。GSM 通信模块 MC35IT 支持 GSM07.05 所定义的 AT 指令集的指令, 具有 GSM 无线通信的全部功能, 能够满足收发短信的要求。将模块与网关相连, 发送和接收 SMS 信息。我们采用 PDU Mode, 所有手机均支持该模式, 可以使用任何字符集, 这也是手机默认的编码方式, 可以实现编码和解码 PDU 格式的短消息。

③以太网 Internet 与家庭信息网络的通信。通过 HTML 语言, 网关作为服务器来实现。嵌入式 Linux 内核保留 TCP/IP 及 HTTP 协议的支持, 可采用 CGI 或者 PHP 等技术编写动态网页, 实现用户控制界面以及对终端设备的控制 。

2.3　信息流服务

信息流服务器是由智能家居环境网关启动的一个单独的服务程序, 它负责沟通家庭内外的信息流通道, 主要是传输音频和视频信号, 要求数据传输速度比较高, 如利用摄像头通过 Internet 可以监视到家庭内部实时的状况, 需要由网关作为信息流服务器实现存储转发功能。

3　家庭安全

本系统的安全机制采用用户验证、操作权限匹配、附加密钥、工作日志管理多种策略。用户的登录实行异用户多权限机制, 不同的用户对应着不同的安全等级和操作权限, 如系统用户可以监控家庭关键设备, 如暖气、煤气、门窗等。而一般用户只能有一般的控制权利, 如电视机、收音机等, 每个用户具有对应的权限等级。登录用户必须先经过身份验证才能登录到系统中, 具有一个合法用户的身份以及对应的权限。根据用户登录的账号和密码, 在数据库中进行查找, 如果登录成功对该用户进行初始化, 并回发用户一个密钥; 否则拒绝登录请求, 断开连接。所有的登录活动都记录到工作日志。用户收到密钥, 按一定规则对密钥进行处理, 得到一串数据码。并把结果数据码作为报头和控制命令一起作为控制命令。智能家居环境网关收到并验证密钥无误后, 只有当用户的权限等级不低于操作命令的权限等级, 该命令才可以执行, 并将该操作相关信息保存到系统的工作日志中。

4　结束语

智能家居环境系统具有广阔的市场应用发展前景, 正向着集成化、智能化、模块化和规模化方向发展。本文介绍的智能家居环境系统采用嵌入式系统开发, 各功能模块的独立性较强, 还具有良好的界面接口、结构明晰, 运用可移动控制服务代理的思想开发智能家居环境网关, 为用户提供良好的控制界面, 并可通过多种可移动控制终端对家庭信息网络进行控制。其进一步的发展尚需要多方面的不断完善, 如家电统一规范的制定、通信协议标准和无线技术等。

参考文献:

[1] 杨士元. 掀开智能家居的面纱[N]. 中国计算机用户, 2002, (33).

[2] 石志国, 王志良. 基于移动 Agent 的 GSM 短消息查询系统[J]. 计算机工程, 2003, 29(7): 49-51.

[3] 李华. 服务代理技术工作机制的分析[J]. 通信与信息技术, 2003, (5): 39-42.

作者简介:

秦勃(1964-), 男, 山东青岛人, 副教授, 在职博士, 研究方向为计算机图形学、计算机控制; 王琳(1977-), 女, 山东荣成人, 助理工程师, 硕士, 研究方向为嵌入式系统; 邵峰晶(1960-), 女, 山东青岛人, 教授, 博士, 研究方向为数据挖掘、嵌入式系统; 於雷(1980-), 男, 安徽人, 硕士, 主要研究方向为嵌入式系统、移动通信。

(上接第 238 页)

[3] Leach. TBSE: An Engineering Approach to the Design of Accurate and Reliable Security Systems[J]. Computer & Security, 2004, 23(1): 265-266.

[4] M Li. An Approach for Reliably Identifying Signs of DDoS Flood Attacks Based on LRD Traffic Pattern Recognition[J]. Computer & Security, 2004, 23.

[5] L Garber. Denial-of-service Attacks Rip the Internet[J]. IEEE Computer, 2000, 33(4): 12-17.

[6] R Bace. An Introduction to Intrusion Detection and Assessment[Z]. ICSA, Inc., 2000.

[7] K Liston. Intrusion Detection FAQ: Can You Explain Traffic Analysis and Anomaly Detection? [EB/OL]. www.sans.org/resources/idfaq/anomaly_detection.php, 2004-06.

[8] S S Kim, A L N Reddy, M Vannucci. Detecting Traffic Anomalies at the Source Though Aggregate Analysis of Packet Header Data[C]. Proc., Networking 2004, LNCS 3042, 2004. 1047-1059.

[9] Manual Reference Pages-PCAP (3)[EB/OL]. http://www.squarebox.co.uk/cgi-squarebox/manServer/usr/share/man/man3/pcap.3, 2004-12-09.

[10] V Hegde. Exploring TCP/IP with TCPdump and Tethereal[EB/OL]. http://www.linuxgazette.com/issue86/vinayak.html, 2004.

[11] S Ostermann. Tcptrace[EB/OL]. http://jarok.cs.ohiou.edu/software/tcptrace/tcptrace.html, 2004.

[12] W R Stevens. TCP/IP Illustrated volume 1, The Protocols [M]. American: Addison-Wesley, 2002.

[13] G R Wright, W Richard Stevens. TCP/IP Illustrated volume 2, the Implementation[M]. American: Addison-Wesley, 2002.

作者简介:

胡文静(1971-), 男, 博士研究生, 研究方向为信号检测、通信与信息处理、网络安全; 李明(1955-), 男, 博导, 研究方向为网络安全; 刘锦高(1949-), 男, 博导, 研究方向为信号检测、通信与信息处理。